



TM32G05x 系列

API 用户手册

版本 1.00

hitenx reserves the right to change or discontinue the manual and online documentation to this product herein to improve reliability, function or design without further notice. hitenx does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. hitenx products are not designed, intended, or authorized for use in life support appliances, devices, or systems. If Buyer purchases or uses hitenx products for any such unintended or unauthorized application, Buyer shall indemnify and hold hitenx and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that hitenx was negligent regarding the design or manufacture of the part.

修改记录

版本	日期	描述
V1.00	2024/02/28	新颁

HITENX

目录

1	概述	63
2	API (基础 LL 层)	64
2.1	IWDG 模块	64
2.1.1	LL_IWDG_Enable	64
2.1.2	LL_IWDG_ReloadCounter	64
2.1.3	LL_IWDG_EnableWriteAccess	65
2.1.4	LL_IWDG_DisableWriteAccess	65
2.1.5	LL_IWDG_SetPrescaler	65
2.1.6	LL_IWDG_GetPrescaler	66
2.1.7	LL_IWDG_SetReloadCounter	66
2.1.8	LL_IWDG_GetReloadCounter	66
2.1.9	LL_IWDG_SetWindow	67
2.1.10	LL_IWDG_GetWindow	67
2.1.11	LL_IWDG_IsActiveFlag_PVU	67
2.1.12	LL_IWDG_IsActiveFlag_RVU	67
2.1.13	LL_IWDG_IsActiveFlag_WVU	68
2.1.14	LL_IWDG_IsActiveFlag	68
2.1.15	LL_IWDG_IsReady	68
2.2	CORTEX 模块	69
2.2.1	LL_SYSTICK_IsActiveCounterFlag	69
2.2.2	LL_SYSTICK_SetClkSource	69
2.2.3	LL_SYSTICK_GetClkSource	69
2.2.4	LL_SYSTICK_EnableIT	70
2.2.5	LL_SYSTICK_DisableIT	70
2.2.6	LL_SYSTICK_IsEnabledIT	70
2.2.7	LL_LPM_EnableSleep	70
2.2.8	LL_LPM_EnableDeepSleep	71
2.2.9	LL_LPM_EnableSleepOnExit	71
2.2.10	LL_LPM_DisableSleepOnExit	71
2.2.11	LL_LPM_IsEnabledSleepOnExit	71
2.2.12	LL_LPM_EnableEventOnPend	72
2.2.13	LL_LPM_DisableEventOnPend	72
2.2.14	LL_CPUID_GetImplementer	72
2.2.15	LL_CPUID_GetVariant	73
2.2.16	LL_CPUID_GetArchitecture	73
2.2.17	LL_CPUID_GetParNo	73
2.2.18	LL_CPUID_GetRevision	73
2.3	EXTI 模块	74
2.3.1	LL_EXTI_EnableIT	74
2.3.2	LL_EXTI_DisableIT	74
2.3.3	LL_EXTI_IsEnabledIT	75
2.3.4	LL_EXTI_EnableEvent	75
2.3.5	LL_EXTI_DisableEvent	76

2.3.6	LL_EXTI_IsEnabledEvent	76
2.3.7	LL_EXTI_EnableRisingTrig	77
2.3.8	LL_EXTI_DisableRisingTrig	77
2.3.9	LL_EXTI_IsEnabledRisingTrig	78
2.3.10	LL_EXTI_EnableFallingTrig	78
2.3.11	LL_EXTI_DisableFallingTrig	78
2.3.12	LL_EXTI_IsEnabledFallingTrig	79
2.3.13	LL_EXTI_IsActiveFallingFlag	79
2.3.14	LL_EXTI_ReadFallingFlag	80
2.3.15	LL_EXTI_ClearFallingFlag	80
2.3.16	LL_EXTI_IsActiveRisingFlag	80
2.3.17	LL_EXTI_ReadRisingFlag	81
2.3.18	LL_EXTI_ClearRisingFlag	81
2.3.19	LL_EXTI_SetEXTISource_0_7	82
2.3.20	LL_EXTI_GetEXTISource_0_7	82
2.3.21	LL_EXTI_SetEXTISource_8_15	82
2.3.22	LL_EXTI_GetEXTISource_8_15	83
2.3.23	LL_EXTI_Init	83
2.3.24	LL_EXTI_DeInit	83
2.3.25	LL_EXTI_StructInit	84
2.4	FLASH 模块	84
2.4.1	LL_FLASH_SetCSConfig	84
2.4.2	LL_FLASH_GetCSConfig	84
2.4.3	LL_FLASH_SetLatency	85
2.4.4	LL_FLASH_GetLatency	85
2.4.5	LL_FLASH_EnablePrefetch	85
2.4.6	LL_FLASH_DisablePrefetch	86
2.4.7	LL_FLASH_IsEnabledPrefetch	86
2.4.8	LL_FLASH_EnableProgRror	86
2.4.9	LL_FLASH_DisableProgRror	86
2.4.10	LL_FLASH_IsEnabledProgRror	87
2.4.11	LL_FLASH_IsActiveFlag	87
2.4.12	LL_FLASH_GetActiveFlag	87
2.4.13	LL_FLASH_ClearFlag	88
2.4.14	LL_FLASH_IsActiveFlag_AccessError	88
2.4.15	LL_FLASH_ClearFlag_AccessError	89
2.4.16	LL_FLASH_IsActiveFlag_ReadError	89
2.4.17	LL_FLASH_ClearFlag_ReadError	89
2.4.18	LL_FLASH_IsActiveFlag_ProgEraseInterrupt	89
2.4.19	LL_FLASH_ClearFlag_ProgEraseInterrupt	90
2.4.20	LL_FLASH_IsActiveFlag_ProgFinished	90
2.4.21	LL_FLASH_ClearFlag_ProgFinished	90
2.4.22	LL_FLASH_IsActiveFlag_Busy	90
2.4.23	LL_FLASH_IsActiveFlag_ProgEraseSeqError	91
2.4.24	LL_FLASH_ClearFlag_ProgEraseSeqError	91

2.4.25	LL_FLASH_IsActiveFlag_ProgMissError.....	91
2.4.26	LL_FLASH_ClearFlag_ProgMissError	92
2.4.27	LL_FLASH_IsActiveFlag_ProgSizeError.....	92
2.4.28	LL_FLASH_ClearFlag_ProgSizeError	92
2.4.29	LL_FLASH_IsActiveFlag_ProgAlignError	92
2.4.30	LL_FLASH_ClearFlag_ProgAlignError	93
2.4.31	LL_FLASH_IsActiveFlag_ProgError	93
2.4.32	LL_FLASH_ClearFlag_ProgError	93
2.4.33	LL_FLASH_IsActiveFlag_WRPError	93
2.4.34	LL_FLASH_ClearFlag_WRPError	94
2.4.35	LL_FLASH_Lock.....	94
2.4.36	LL_FLASH_Unlock	94
2.4.37	LL_FLASH_IsLocked.....	95
2.4.38	LL_FLASH_OB_Lock	95
2.4.39	LL_FLASH_OB_Unlock.....	95
2.4.40	LL_FLASH_OB_IsLocked.....	95
2.4.41	LL_FLASH_EnableSecurityProtect	96
2.4.42	LL_FLASH_IsEnabledSecurityProtect	96
2.4.43	LL_FLASH_OB_Launch	96
2.4.44	LL_FLASH_OB_Launch_IsReady	96
2.4.45	LL_FLASH_EnableIT_ReadError	97
2.4.46	LL_FLASH_DisableIT_ReadError	97
2.4.47	LL_FLASH_IsEnabledIT_ReadError.....	97
2.4.48	LL_FLASH_EnableIT_ProgramError.....	98
2.4.49	LL_FLASH_DisableIT_ProgramError.....	98
2.4.50	LL_FLASH_IsEnabledIT_ProgramError	98
2.4.51	LL_FLASH_EnableIT_ProgramFinish	98
2.4.52	LL_FLASH_DisableIT_ProgramFinish	99
2.4.53	LL_FLASH_IsEnabledIT_ProgramFinish	99
2.4.54	LL_FLASH_EnableIT	99
2.4.55	LL_FLASH_DisableIT.....	99
2.4.56	LL_FLASH_IsEnabledIT	100
2.4.57	LL_FLASH_SetErasePageN	100
2.4.58	LL_FLASH_GetErasePageN.....	100
2.4.59	LL_FLASH_OB_StartReload.....	101
2.4.60	LL_FLASH_OBReloading_IsOngoing	101
2.4.61	LL_FLASH_EraseStart	101
2.4.62	LL_FLASH_Erase_IsOngoing	101
2.4.63	LL_FLASH_SetEraseMode.....	102
2.4.64	LL_FLASH_GetEraseMode	102
2.4.65	LL_FLASH_SetProgramMode.....	102
2.4.66	LL_FLASH_GetProgramMode	103
2.4.67	LL_FLASH_OB_SetBORHysteresis	103
2.4.68	LL_FLASH_OB_GetBORHysteresis.....	103
2.4.69	LL_FLASH_OB_SetBORLevel.....	104
2.4.70	LL_FLASH_OB_GetBORLevel	104

2.4.71	LL_FLASH_OB_SetBORState	104
2.4.72	LL_FLASH_OB_GetBORState	105
2.4.73	LL_FLASH_OB_SetRDPLLevel	105
2.4.74	LL_FLASH_OB_GetRDPLLevel.....	105
2.4.75	LL_FLASH_OB_SetBOOTSEL	105
2.4.76	LL_FLASH_OB_GetBOOTSEL.....	106
2.4.77	LL_FLASH_OB_SetBOOT0SW	106
2.4.78	LL_FLASH_OB_GetBOOT0SW.....	106
2.4.79	LL_FLASH_OB_SetResetPin	107
2.4.80	LL_FLASH_OB_GetResetPin	107
2.4.81	LL_FLASH_OB_SetBootSize.....	107
2.4.82	LL_FLASH_OB_GetBootSize	108
2.4.83	LL_FLASH_OB_SetIWDGTypeInStopMode.....	108
2.4.84	LL_FLASH_OB_GetIWDGTypeInStopMode	108
2.4.85	LL_FLASH_OB_SetIWDGSourceType	108
2.4.86	LL_FLASH_OB_GetIWDGSourceType.....	109
2.4.87	LL_FLASH_OB_SetPCROPASStartPage.....	109
2.4.88	LL_FLASH_OB_GetPCROPASStartPage	109
2.4.89	LL_FLASH_OB_SetPCROPAEndPage.....	110
2.4.90	LL_FLASH_OB_GetPCROPAEndPage	110
2.4.91	LL_FLASH_OB_SetPCROPBStartPage.....	110
2.4.92	LL_FLASH_OB_GetPCROPBStartPage	110
2.4.93	LL_FLASH_OB_SetPCROPBEndPage.....	111
2.4.94	LL_FLASH_OB_GetPCROPBEndPage	111
2.4.95	LL_FLASH_OB_SetWRPASStartPage.....	111
2.4.96	LL_FLASH_OB_GetWRPASStartPage	112
2.4.97	LL_FLASH_OB_SetWRPAEndPage.....	112
2.4.98	LL_FLASH_OB_GetWRPAEndPage	112
2.4.99	LL_FLASH_OB_SetWRPBStartPage.....	112
2.4.100	LL_FLASH_OB_GetWRPBStartPage	113
2.4.101	LL_FLASH_OB_SetWRPBEndPage.....	113
2.4.102	LL_FLASH_OB_GetWRPBEndPage	113
2.4.103	LL_FLASH_OB_SetSecSize.....	114
2.4.104	LL_FLASH_OB_GetSecSize	114
2.4.105	LL_FLASH_OB_SetBootLock	114
2.4.106	LL_FLASH_OB_GetBootLock.....	114
2.5	GPIO 模块	115
2.5.1	LL_GPIO_SetPinMode	116
2.5.2	LL_GPIO_GetPinMode.....	116
2.5.3	LL_GPIO_SetPinOutputType.....	117
2.5.4	LL_GPIO_GetPinOutputType	117
2.5.5	LL_GPIO_SetPinPull	118
2.5.6	LL_GPIO_GetPinPull.....	118
2.5.7	LL_GPIO_SetAFPin_0_7.....	119
2.5.8	LL_GPIO_GetAFPin_0_7	119

2.5.9	LL_GPIO_SetAFPin_8_15.....	119
2.5.10	LL_GPIO_GetAFPin_8_15.....	120
2.5.11	LL_GPIO_LockPin.....	120
2.5.12	LL_GPIO_IsPinLocked.....	121
2.5.13	LL_GPIO_IsAnyPinLocked.....	121
2.5.14	LL_GPIO_ReadInputPort.....	121
2.5.15	LL_GPIO_IsInputPinSet.....	122
2.5.16	LL_GPIO_WriteOutputPort.....	122
2.5.17	LL_GPIO_ReadOutputPort.....	122
2.5.18	LL_GPIO_IsOutputPinSet.....	123
2.5.19	LL_GPIO_SetOutputPin.....	123
2.5.20	LL_GPIO_ResetOutputPin.....	124
2.5.21	LL_GPIO_TogglePin.....	124
2.5.22	LL_GPIO_DeInit.....	124
2.5.23	LL_GPIO_Init.....	125
2.5.24	LL_GPIO_StructInit.....	125
2.6	PWR 模块.....	125
2.6.1	LL_PWR_EnableBGR.....	125
2.6.2	LL_PWR_DisableBGR.....	126
2.6.3	LL_PWR_IsEnabledBGR.....	126
2.6.4	LL_PWR_SetVref.....	126
2.6.5	LL_PWR_GetVref.....	126
2.6.6	LL_PWR_EnableBORPORIntermittentDetection.....	127
2.6.7	LL_PWR_DisableBORPORIntermittentDetection.....	127
2.6.8	LL_PWR_IsEnabledBORPORIntermittentDetection.....	127
2.6.9	LL_PWR_EnableLowPowerRunMode.....	128
2.6.10	LL_PWR_DisableLowPowerRunMode.....	128
2.6.11	LL_PWR_IsEnabledLowPowerRunMode.....	128
2.6.12	LL_PWR_SetPowerMode.....	128
2.6.13	LL_PWR_GetPowerMode.....	129
2.6.14	LL_PWR_PVD_SetLevel.....	129
2.6.15	LL_PWR_PVD_GetLevel.....	130
2.6.16	LL_PWR_PVD_SetSelection.....	130
2.6.17	LL_PWR_PVD_GetSelection.....	130
2.6.18	LL_PWR_PVD_SetHysteresis.....	131
2.6.19	LL_PWR_PVD_GetHysteresis.....	131
2.6.20	LL_PWR_PVD_SetFilterTime.....	131
2.6.21	LL_PWR_PVD_GetFilterTime.....	132
2.6.22	LL_PWR_PVD_Enable.....	132
2.6.23	LL_PWR_PVD_Disable.....	132
2.6.24	LL_PWR_PVD_IsEnabled.....	133
2.6.25	LL_PWR_PVD_EnableIT.....	133
2.6.26	LL_PWR_PVD_DisableIT.....	133
2.6.27	LL_PWR_PVD_IsEnabledIT.....	133
2.6.28	LL_PWR_PVD_EnableThreshold.....	134

2.6.29	LL_PWR_PVD_DisableThreshold.....	134
2.6.30	LL_PWR_PVD_IsEnabledThreshold.....	134
2.6.31	LL_PWR_PVD_EnableFilter.....	135
2.6.32	LL_PWR_PVD_DisableFilter.....	135
2.6.33	LL_PWR_PVD_IsEnabledFilter.....	135
2.6.34	LL_PWR_PVD_IsActiveFlagIT.....	136
2.6.35	LL_PWR_PVD_ClearFlagIT.....	136
2.6.36	LL_PWR_PVD_GetStatus.....	136
2.6.37	LL_PWR_DeInit.....	136
2.7	RTC 模块.....	137
2.7.1	LL_RTC_SetHourFormat.....	138
2.7.2	LL_RTC_GetHourFormat.....	138
2.7.3	LL_RTC_SetPeriodSource.....	138
2.7.4	LL_RTC_GetPeriodSource.....	139
2.7.5	LL_RTC_SetPRDSTime.....	139
2.7.6	LL_RTC_GetPRDSTime.....	139
2.7.7	LL_RTC_SetPRDXTime.....	140
2.7.8	LL_RTC_GetPRDXTime.....	140
2.7.9	LL_RTC_Set1HZOutputPrecision.....	140
2.7.10	LL_RTC_Get1HZOutputPrecision.....	141
2.7.11	LL_RTC_EnableReadWriteMode.....	141
2.7.12	LL_RTC_DisableReadWriteMode.....	141
2.7.13	LL_RTC_IsEnabledReadWriteMode.....	142
2.7.14	LL_RTC_Enable1HzOutput.....	142
2.7.15	LL_RTC_Disable1HzOutput.....	142
2.7.16	LL_RTC_IsEnabled1HzOutput.....	142
2.7.17	LL_RTC_Enable.....	143
2.7.18	LL_RTC_Disable.....	143
2.7.19	LL_RTC_IsEnabled.....	143
2.7.20	LL_RTC_TIME_SetHour.....	143
2.7.21	LL_RTC_TIME_GetHour.....	144
2.7.22	LL_RTC_TIME_SetMinute.....	144
2.7.23	LL_RTC_TIME_GetMinute.....	144
2.7.24	LL_RTC_TIME_SetSecond.....	145
2.7.25	LL_RTC_TIME_GetSecond.....	145
2.7.26	LL_RTC_TIME_Config.....	145
2.7.27	LL_RTC_TIME_Get.....	146
2.7.28	LL_RTC_DATE_SetYear.....	146
2.7.29	LL_RTC_DATE_GetYear.....	146
2.7.30	LL_RTC_DATE_SetWeekDay.....	147
2.7.31	LL_RTC_DATE_GetWeekDay.....	147
2.7.32	LL_RTC_DATE_SetMonth.....	147
2.7.33	LL_RTC_DATE_GetMonth.....	148
2.7.34	LL_RTC_DATE_SetDay.....	148
2.7.35	LL_RTC_DATE_GetDay.....	148

2.7.36	LL_RTC_DATE_Config.....	149
2.7.37	LL_RTC_DATE_Get.....	149
2.7.38	LL_RTC_ALM_Enable.....	150
2.7.39	LL_RTC_ALM_Disable.....	150
2.7.40	LL_RTC_ALM_IsEnabled.....	150
2.7.41	LL_RTC_ALM_SetWeekDay.....	150
2.7.42	LL_RTC_ALM_GetWeekDay.....	151
2.7.43	LL_RTC_ALM_SetHour.....	151
2.7.44	LL_RTC_ALM_GetHour.....	152
2.7.45	LL_RTC_ALM_SetMinute.....	152
2.7.46	LL_RTC_ALM_GetMinute.....	152
2.7.47	LL_RTC_ALM_ConfigTime.....	153
2.7.48	LL_RTC_SetCompensateValue.....	153
2.7.49	LL_RTC_GetCompensateValue.....	153
2.7.50	LL_RTC_EnableCompensate.....	154
2.7.51	LL_RTC_DisableCompensate.....	154
2.7.52	LL_RTC_IsEnabledCompensate.....	154
2.7.53	LL_RTC_IsActiveFlag_ALM.....	155
2.7.54	LL_RTC_IsActiveFlag_PRD.....	155
2.7.55	LL_RTC_ClearFlag_ALM.....	155
2.7.56	LL_RTC_ClearFlag_PRD.....	155
2.7.57	LL_RTC_ALM_EnableIT.....	156
2.7.58	LL_RTC_ALM_DisableIT.....	156
2.7.59	LL_RTC_ALM_IsEnabledIT.....	156
2.7.60	LL_RTC_DeInit.....	156
2.7.61	LL_RTC_Init.....	157
2.7.62	LL_RTC_StructInit.....	157
2.7.63	LL_RTC_TIME_Init.....	157
2.7.64	LL_RTC_TIME_StructInit.....	158
2.7.65	LL_RTC_DATE_Init.....	158
2.7.66	LL_RTC_DATE_StructInit.....	158
2.7.67	LL_RTC_ALM_Init.....	159
2.7.68	LL_RTC_ALM_StructInit.....	159
2.7.69	LL_RTC_EnterReadWriteMode.....	159
2.7.70	LL_RTC_ExitReadWriteMode.....	159
2.8	ADC 模块.....	160
2.8.1	LL_ADC_DMA_GetRegAddr.....	161
2.8.2	LL_ADC_SetClock.....	161
2.8.3	LL_ADC_GetClock.....	161
2.8.4	LL_ADC_SetPathInternalCh.....	162
2.8.5	LL_ADC_GetPathInternalCh.....	162
2.8.6	LL_ADC_EnableChBuf.....	163
2.8.7	LL_ADC_DisableChBuf.....	163
2.8.8	LL_ADC_IsEnabledChBuf.....	163
2.8.9	LL_ADC_SetVREF.....	163

2.8.10	LL_ADC_GetVREF	164
2.8.11	LL_ADC_SetCalibrationFactor	164
2.8.12	LL_ADC_GetCalibrationFactor	164
2.8.13	LL_ADC_SetCalibrationGainK	165
2.8.14	LL_ADC_GetCalibrationGainK	165
2.8.15	LL_ADC_SetCalibrationGainB	165
2.8.16	LL_ADC_GetCalibrationGainB	166
2.8.17	LL_ADC_SetCalibrationTimes	166
2.8.18	LL_ADC_GetCalibrationTimes	166
2.8.19	LL_ADC_SetDataAlignment	166
2.8.20	LL_ADC_GetDataAlignment	167
2.8.21	LL_ADC_SetLowPowerMode	167
2.8.22	LL_ADC_GetLowPowerMode	167
2.8.23	LL_ADC_REG_SetTriggerSource	168
2.8.24	LL_ADC_REG_GetTriggerSource	168
2.8.25	LL_ADC_REG_SetTriggerEdge	169
2.8.26	LL_ADC_REG_GetTriggerEdge	169
2.8.27	LL_ADC_REG_SetSequencerLength	169
2.8.28	LL_ADC_REG_GetSequencerLength	170
2.8.29	LL_ADC_REG_SetSequencerRanks	170
2.8.30	LL_ADC_REG_GetSequencerRanks	171
2.8.31	LL_ADC_REG_SetConversionMode	172
2.8.32	LL_ADC_REG_GetConversionMode	173
2.8.33	LL_ADC_REG_SetDMATransfer	173
2.8.34	LL_ADC_REG_GetDMATransfer	173
2.8.35	LL_ADC_REG_SetOverrun	174
2.8.36	LL_ADC_REG_GetOverrun	174
2.8.37	LL_ADC_SetChannelSamplingTime	174
2.8.38	LL_ADC_GetChannelSamplingTime	175
2.8.39	LL_ADC_AnalogWDG_Enable	175
2.8.40	LL_ADC_AnalogWDG_Disable	176
2.8.41	LL_ADC_AnalogWDG_IsEnabled	176
2.8.42	LL_ADC_ConfigAnalogWDThresholds	176
2.8.43	LL_ADC_SetAnalogWDThresholds	177
2.8.44	LL_ADC_GetAnalogWDThresholds	177
2.8.45	LL_ADC_Enable	177
2.8.46	LL_ADC_Disable	178
2.8.47	LL_ADC_IsEnabled	178
2.8.48	LL_ADC_IsDisableOngoing	178
2.8.49	LL_ADC_StartCalibration	178
2.8.50	LL_ADC_IsCalibrationOnGoing	179
2.8.51	LL_ADC_REG_StartConversion	179
2.8.52	LL_ADC_REG_StopConversion	179
2.8.53	LL_ADC_REG_IsConversionOngoing	179
2.8.54	LL_ADC_REG_IsStopConversionOngoing	180
2.8.55	LL_ADC_REG_ReadConversionData32	180

2.8.56	LL_ADC_REG_ReadConversionData12	180
2.8.57	LL_ADC_IsActiveFlag_ADRDY	181
2.8.58	LL_ADC_IsActiveFlag_EOC.....	181
2.8.59	LL_ADC_IsActiveFlag_EOS	181
2.8.60	LL_ADC_IsActiveFlag_OVR	181
2.8.61	LL_ADC_IsActiveFlag_AWDG	182
2.8.62	LL_ADC_IsActiveFlag_EOCAL	182
2.8.63	LL_ADC_IsActiveFlag	182
2.8.64	LL_ADC_ClearFlag_ADRDY	183
2.8.65	LL_ADC_ClearFlag_EOC	183
2.8.66	LL_ADC_ClearFlag_EOS.....	183
2.8.67	LL_ADC_ClearFlag_OVR.....	183
2.8.68	LL_ADC_ClearFlag_AWD1	184
2.8.69	LL_ADC_ClearFlag_EOCAL	184
2.8.70	LL_ADC_ClearFlag	184
2.8.71	LL_ADC_EnableIT_ADRDY	185
2.8.72	LL_ADC_EnableIT_EOC	185
2.8.73	LL_ADC_EnableIT_EOS.....	185
2.8.74	LL_ADC_EnableIT_OVR.....	185
2.8.75	LL_ADC_EnableIT_AWD1	186
2.8.76	LL_ADC_EnableIT_EOCAL	186
2.8.77	LL_ADC_EnableIT	186
2.8.78	LL_ADC_DisableIT_ADRDY	187
2.8.79	LL_ADC_DisableIT_EOC	187
2.8.80	LL_ADC_DisableIT_EOS.....	187
2.8.81	LL_ADC_DisableIT_OVR.....	187
2.8.82	LL_ADC_DisableIT_AWD1	188
2.8.83	LL_ADC_DisableIT_EOCAL	188
2.8.84	LL_ADC_DisableIT	188
2.8.85	LL_ADC_IsEnabledIT_ADRDY	189
2.8.86	LL_ADC_IsEnabledIT_EOC	189
2.8.87	LL_ADC_IsEnabledIT_EOS.....	189
2.8.88	LL_ADC_IsEnabledIT_OVR.....	189
2.8.89	LL_ADC_IsEnabledIT_AWD1	190
2.8.90	LL_ADC_IsEnabledIT_EOCAL	190
2.8.91	LL_ADC_IsEnabledIT	190
2.8.92	LL_ADC_DeInit.....	191
2.8.93	LL_ADC_Init	191
2.8.94	LL_ADC_StructInit.....	191
2.8.95	LL_ADC_REG_Init.....	191
2.8.96	LL_ADC_REG_StructInit	192
2.9	ATK 模块.....	192
2.9.1	LL_ATK_SetClockFrequency	193
2.9.2	LL_ATK_GetClockFrequency.....	194
2.9.3	LL_ATK_EnableClockFrequencyAutoSet	194

2.9.4	LL_ATK_DisableClockFrequencyAutoSet	194
2.9.5	LL_ATK_IsEnabledClockFrequencyAutoSet	195
2.9.6	LL_ATK_SetScanningLength	195
2.9.7	LL_ATK_GetScanningLength	195
2.9.8	LL_ATK_EnableExternalCap	196
2.9.9	LL_ATK_DisableExternalCap	196
2.9.10	LL_ATK_IsEnabledExternalCap	196
2.9.11	LL_ATK_EnableAutoRerun	197
2.9.12	LL_ATK_DisableAutoRerun	197
2.9.13	LL_ATK_IsEnabledAutoRerun	197
2.9.14	LL_ATK_SetScanningMode	198
2.9.15	LL_ATK_GetScanningMode	198
2.9.16	LL_ATK_SetVoltageReference	198
2.9.17	LL_ATK_GetVoltageReference	199
2.9.18	LL_ATK_SetChannelTrimming	199
2.9.19	LL_ATK_GetChannelTrimming	200
2.9.20	LL_ATK_EnableClockfrequencySpread	200
2.9.21	LL_ATK_DisableClockfrequencySpread	201
2.9.22	LL_ATK_IsEnabledClockfrequencySpread	201
2.9.23	LL_ATK_EnableIT_Finished	201
2.9.24	LL_ATK_DisableIT_Finished	202
2.9.25	LL_ATK_IsEnabledIT_Finished	202
2.9.26	LL_ATK_IsActiveFlag_EOC	202
2.9.27	LL_ATK_IsActiveFlag_IT	203
2.9.28	LL_ATK_ClearFlag_IT	203
2.9.29	LL_ATK_EnableChannel	203
2.9.30	LL_ATK_DisableChannel	204
2.9.31	LL_ATK_IsEnabledChannel	204
2.9.32	LL_ATK_Set1stChannel	205
2.9.33	LL_ATK_Get1stChannel	206
2.9.34	LL_ATK_EnableABPinExchange	206
2.9.35	LL_ATK_DisableABPinExchange	207
2.9.36	LL_ATK_IsEnabledABPinExchange	207
2.9.37	LL_ATK_Enable	208
2.9.38	LL_ATK_Disable	208
2.9.39	LL_ATK_IsEnabled	208
2.9.40	LL_ATK_EnableDMA	209
2.9.41	LL_ATK_DisableDMA	209
2.9.42	LL_ATK_IsEnabledDMA	209
2.9.43	LL_ATK_GetRAMValue	210
2.9.44	LL_ATK_GetRegAddr	210
2.9.45	LL_ATK_EnableShield	210
2.9.46	LL_ATK_DisableShield	211
2.9.47	LL_ATK_IsEnabledShield	211
2.9.48	LL_ATK_EnableConvert	211
2.9.49	LL_ATK_DisableConvert	212

2.9.50	LL_ATK_IsEnabledConvert	212
2.10	BUS 模块	212
2.10.1	LL_AHB1_GRP1_EnableClock	212
2.10.2	LL_AHB1_GRP1_IsEnabledClock	213
2.10.3	LL_AHB1_GRP1_DisableClock	213
2.10.4	LL_AHB1_GRP1_ForceReset	213
2.10.5	LL_AHB1_GRP1_ReleaseReset	214
2.10.6	LL_APB1_GRP1_EnableClock	214
2.10.7	LL_APB1_GRP1_IsEnabledClock	214
2.10.8	LL_APB1_GRP1_DisableClock	215
2.10.9	LL_APB1_GRP1_ForceReset	215
2.10.10	LL_APB1_GRP1_ReleaseReset	216
2.10.11	LL_APB2_GRP1_EnableClock	216
2.10.12	LL_APB2_GRP1_IsEnabledClock	217
2.10.13	LL_APB2_GRP1_DisableClock	217
2.10.14	LL_APB2_GRP1_ForceReset	217
2.10.15	LL_APB2_GRP1_ReleaseReset	218
2.10.16	LL_IOP_GRP1_EnableClock	218
2.10.17	LL_IOP_GRP1_IsEnabledClock	218
2.10.18	LL_IOP_GRP1_DisableClock	219
2.10.19	LL_IOP_GRP1_ForceReset	219
2.10.20	LL_IOP_GRP1_ReleaseReset	219
2.11	COMP 模块	220
2.11.1	LL_COMP_ConfigInputs	220
2.11.2	LL_COMP_SetInputPlus	221
2.11.3	LL_COMP_GetInputPlus	221
2.11.4	LL_COMP_SetInputMinus	221
2.11.5	LL_COMP_GetInputMinus	222
2.11.6	LL_COMP_SetWindowMode	222
2.11.7	LL_COMP_GetWindowMode	223
2.11.8	LL_COMP_SetOutputPolarity	223
2.11.9	LL_COMP_GetOutputPolarity	223
2.11.10	LL_COMP_SetWindowOutput	224
2.11.11	LL_COMP_GetWindowOutput	224
2.11.12	LL_COMP_FILTER_Enable	224
2.11.13	LL_COMP_FILTER_Disable	224
2.11.14	LL_COMP_FILTER_IsEnabled	225
2.11.15	LL_COMP_SetOutputFilterTimes	225
2.11.16	LL_COMP_GetOutputFilterTimes	226
2.11.17	LL_COMP_Enable	226
2.11.18	LL_COMP_Disable	226
2.11.19	LL_COMP_IsEnabled	227
2.11.20	LL_COMP_Lock	227
2.11.21	LL_COMP_IsLocked	227
2.11.22	LL_COMP_ReadOutputLevel	227

2.11.23	LL_COMP_DeInit	228
2.11.24	LL_COMP_Init.....	228
2.11.25	LL_COMP_StructInit	228
2.12	CRC 模块.....	229
2.12.1	LL_CRC_SetPolynomialSize	229
2.12.2	LL_CRC_GetPolynomialSize.....	229
2.12.3	LL_CRC_SetInitialData	229
2.12.4	LL_CRC_GetInitialData.....	230
2.12.5	LL_CRC_FeedData32	230
2.12.6	LL_CRC_FeedData16	230
2.12.7	LL_CRC_FeedData8	231
2.12.8	LL_CRC_ReadData32.....	231
2.12.9	LL_CRC_ReadData16.....	231
2.12.10	LL_CRC_ReadData8.....	231
2.12.11	LL_CRC_IsActiveFlag_ResultOK	232
2.12.12	LL_CRC_DeInit	232
2.12.13	LL_CRC_Init	232
2.12.14	LL_CRC_StructInit	233
2.12.15	LL_CRC_Accumulate	233
2.12.16	LL_CRC_Calculate	233
2.13	DAC 模块.....	234
2.13.1	LL_DAC_SetSVREF.....	235
2.13.2	LL_DAC_GetSVREF	236
2.13.3	LL_DAC_SetTriggerSource	236
2.13.4	LL_DAC_GetTriggerSource	237
2.13.5	LL_DAC_SetWaveAutoGeneration	237
2.13.6	LL_DAC_GetWaveAutoGeneration.....	238
2.13.7	LL_DAC_SetWaveNoiseLFSR	238
2.13.8	LL_DAC_GetWaveNoiseLFSR.....	239
2.13.9	LL_DAC_SetWaveTriangleAmplitude.....	239
2.13.10	LL_DAC_GetWaveTriangleAmplitude	240
2.13.11	LL_DAC_SetOutputBuffer.....	240
2.13.12	LL_DAC_GetOutputBuffer.....	241
2.13.13	LL_DAC_SetOutputConnection.....	241
2.13.14	LL_DAC_GetOutputConnection	242
2.13.15	LL_DAC_SetOutputBufferMode	242
2.13.16	LL_DAC_GetOutputBufferMode.....	243
2.13.17	LL_DAC_SetOutputBufferCalibrationLSI.....	243
2.13.18	LL_DAC_GetOutputBufferCalibrationLSI	243
2.13.19	LL_DAC_SetOutputBufferGain	244
2.13.20	LL_DAC_GetOutputBufferGain	244
2.13.21	LL_DAC_EnableDMAReq	244
2.13.22	LL_DAC_DisableDMAReq	245
2.13.23	LL_DAC_IsDMAReqEnabled	245
2.13.24	LL_DAC_DMA_GetRegAddr	246

2.13.25	LL_DAC_Enable.....	246
2.13.26	LL_DAC_Disable.....	246
2.13.27	LL_DAC_IsEnabled.....	247
2.13.28	LL_DAC_EnableTrigger.....	247
2.13.29	LL_DAC_DisableTrigger.....	247
2.13.30	LL_DAC_IsEnabledTrigger.....	248
2.13.31	LL_DAC_TrigSWConversion.....	248
2.13.32	LL_DAC_ConvertData12RightAligned.....	248
2.13.33	LL_DAC_ConvertData12LeftAligned.....	249
2.13.34	LL_DAC_ConvertData8RightAligned.....	249
2.13.35	LL_DAC_ConvertDualData12RightAligned.....	250
2.13.36	LL_DAC_ConvertDualData12LeftAligned.....	250
2.13.37	LL_DAC_ConvertDualData8RightAligned.....	250
2.13.38	LL_DAC_RetrieveOutputData.....	251
2.13.39	LL_DAC_IsActiveFlag_DMAUDR.....	251
2.13.40	LL_DAC_ClearFlag_DMAUDR.....	251
2.13.41	LL_DAC_EnableIT_DMAUDR.....	252
2.13.42	LL_DAC_DisableIT_DMAUDR.....	252
2.13.43	LL_DAC_IsEnabledIT_DMAUDR.....	252
2.13.44	LL_DAC_DeInit.....	253
2.13.45	LL_DAC_Init.....	253
2.13.46	LL_DAC_StructInit.....	253
2.14	DMA 模块.....	254
2.14.1	LL_DMA_EnableChannel.....	256
2.14.2	LL_DMA_DisableChannel.....	256
2.14.3	LL_DMA_IsEnabledChannel.....	256
2.14.4	LL_DMA_SetSoftwareRequire.....	257
2.14.5	LL_DMA_GetSoftwareRequire.....	257
2.14.6	LL_DMA_ConfigTransfer.....	258
2.14.7	LL_DMA_SetMode.....	258
2.14.8	LL_DMA_GetMode.....	259
2.14.9	LL_DMA_SetSourceIncMode.....	259
2.14.10	LL_DMA_GetSourceIncMode.....	259
2.14.11	LL_DMA_SetDestinationIncMode.....	260
2.14.12	LL_DMA_GetDestinationIncMode.....	260
2.14.13	LL_DMA_SetDataSize.....	261
2.14.14	LL_DMA_GetDataSize.....	261
2.14.15	LL_DMA_SetChannelPriorityLevel.....	261
2.14.16	LL_DMA_GetChannelPriorityLevel.....	262
2.14.17	LL_DMA_SetDataLength.....	262
2.14.18	LL_DMA_GetDataLength.....	263
2.14.19	LL_DMA_ConfigAddresses.....	263
2.14.20	LL_DMA_SetSourceAddress.....	263
2.14.21	LL_DMA_SetDestinationAddress.....	264
2.14.22	LL_DMA_GetSourceAddress.....	264

2.14.23	LL_DMA_GetDestinationAddress	265
2.14.24	LL_DMA_IsActiveFlag_GI	265
2.14.25	LL_DMA_IsActiveFlag_TC.....	265
2.14.26	LL_DMA_IsActiveFlag_HT	266
2.14.27	LL_DMA_IsActiveFlag_TE.....	266
2.14.28	LL_DMA_ClearFlag_GI	266
2.14.29	LL_DMA_ClearFlag_TC	267
2.14.30	LL_DMA_ClearFlag_HT	267
2.14.31	LL_DMA_ClearFlag_TE.....	267
2.14.32	LL_DMA_EnableIT_TC	268
2.14.33	LL_DMA_EnableIT_HT	268
2.14.34	LL_DMA_EnableIT_TE.....	268
2.14.35	LL_DMA_DisableIT_TC	269
2.14.36	LL_DMA_DisableIT_HT	269
2.14.37	LL_DMA_DisableIT_TE	269
2.14.38	LL_DMA_IsEnabledIT_TC	270
2.14.39	LL_DMA_IsEnabledIT_HT	270
2.14.40	LL_DMA_IsEnabledIT_TE.....	271
2.14.41	LL_DMA_Init.....	271
2.14.42	LL_DMA_DeInit	271
2.14.43	LL_DMA_StructInit	272
2.14.44	LL_DMAMUX_SetRequestID.....	272
2.14.45	LL_DMAMUX_GetRequestID.....	273
2.14.46	LL_DMAMUX_DisableEventGeneration.....	275
2.14.47	LL_DMAMUX_EnableEventGeneration.....	275
2.14.48	LL_DMAMUX_IsEnabledEventGeneration	276
2.14.49	LL_DMAMUX_SetEventGenerationNumber.....	276
2.14.50	LL_DMAMUX_GetEventGenerationNumber	276
2.14.51	LL_DMAMUX_EnableRequestGen	277
2.14.52	LL_DMAMUX_DisableRequestGen	277
2.14.53	LL_DMAMUX_IsEnabledRequestGen.....	277
2.14.54	LL_DMAMUX_SetRequestGenPolarity.....	278
2.14.55	LL_DMAMUX_GetRequestGenPolarity	278
2.14.56	LL_DMAMUX_SetGenRequestNb.....	278
2.14.57	LL_DMAMUX_GetGenRequestNb.....	279
2.14.58	LL_DMAMUX_SetRequestSignalID.....	279
2.14.59	LL_DMAMUX_GetRequestSignalID.....	280
2.14.60	LL_DMAMUX_IsActiveFlag_RGO	280
2.14.61	LL_DMAMUX_ClearFlag_RGO.....	281
2.14.62	LL_DMAMUX_EnableIT_RGO.....	281
2.14.63	LL_DMAMUX_DisableIT_RGO.....	281
2.14.64	LL_DMAMUX_IsEnabledIT_RGO.....	282
2.15	HIDV 模块.....	282
2.15.1	LL_HDIV_GetDividend.....	282
2.15.2	LL_HDIV_SetDividend.....	282

2.15.3	LL_HDIV_GetDivisor.....	282
2.15.4	LL_HDIV_SetDivisor	283
2.15.5	LL_HDIV_GetSign	283
2.15.6	LL_HDIV_SetSign	283
2.15.7	LL_HDIV_IsActiveFlag_DivisorZero.....	284
2.15.8	LL_HDIV_IsActiveFlag_Finished	284
2.15.9	LL_HDIV_GetQuotient.....	284
2.15.10	LL_HDIV_GetRemainder	284
2.16	I2C 模块.....	285
2.16.1	LL_I2C_DMA_GetRegTxAddr	285
2.16.2	LL_I2C_DMA_GetRegRxAddr	286
2.16.3	LL_I2C_Enable	286
2.16.4	LL_I2C_Disable	286
2.16.5	LL_I2C_IsEnabled.....	286
2.16.6	LL_I2C_SetDigitalFilter.....	287
2.16.7	LL_I2C_GetDigitalFilter.....	287
2.16.8	LL_I2C_EnableDMAReq_TX	287
2.16.9	LL_I2C_DisableDMAReq_TX	288
2.16.10	LL_I2C_IsEnabledDMAReq_TX.....	288
2.16.11	LL_I2C_EnableDMAReq_RX	288
2.16.12	LL_I2C_DisableDMAReq_RX	288
2.16.13	LL_I2C_IsEnabledDMAReq_RX	289
2.16.14	LL_I2C_DMA_GetRegAddr.....	289
2.16.15	LL_I2C_EnableClockStretching.....	289
2.16.16	LL_I2C_DisableClockStretching	290
2.16.17	LL_I2C_IsEnabledClockStretching.....	290
2.16.18	LL_I2C_EnableSlaveByteControl	290
2.16.19	LL_I2C_DisableSlaveByteControl.....	290
2.16.20	LL_I2C_IsEnabledSlaveByteControl	291
2.16.21	LL_I2C_EnableWakeUpFromStop.....	291
2.16.22	LL_I2C_DisableWakeUpFromStop.....	291
2.16.23	LL_I2C_IsEnabledWakeUpFromStop.....	291
2.16.24	LL_I2C_EnableGeneralCall	292
2.16.25	LL_I2C_DisableGeneralCall	292
2.16.26	LL_I2C_IsEnabledGeneralCall	292
2.16.27	LL_I2C_SetMasterAddressingMode.....	293
2.16.28	LL_I2C_GetMasterAddressingMode	293
2.16.29	LL_I2C_SetOwnAddress1.....	293
2.16.30	LL_I2C_GetOwnAddress1	294
2.16.31	LL_I2C_EnableOwnAddress1	294
2.16.32	LL_I2C_DisableOwnAddress1	294
2.16.33	LL_I2C_IsEnabledOwnAddress1	294
2.16.34	LL_I2C_SetOwnAddress2.....	295
2.16.35	LL_I2C_GetOwnAddress2.....	295
2.16.36	LL_I2C_EnableOwnAddress2.....	295

2.16.37	LL_I2C_DisableOwnAddress2.....	296
2.16.38	LL_I2C_IsEnabledOwnAddress2.....	296
2.16.39	LL_I2C_SetTiming.....	296
2.16.40	LL_I2C_GetTimingPrescaler.....	297
2.16.41	LL_I2C_SetTimingPrescaler.....	297
2.16.42	LL_I2C_GetClockLowPeriod.....	297
2.16.43	LL_I2C_SetClockLowPeriod.....	297
2.16.44	LL_I2C_GetClockHighPeriod.....	298
2.16.45	LL_I2C_SetClockHighPeriod.....	298
2.16.46	LL_I2C_GetDataHoldTime.....	298
2.16.47	LL_I2C_SetDataHoldTime.....	299
2.16.48	LL_I2C_GetDataSetupTime.....	299
2.16.49	LL_I2C_SetDataSetupTime.....	299
2.16.50	LL_I2C_EnableIT_TX.....	300
2.16.51	LL_I2C_DisableIT_TX.....	300
2.16.52	LL_I2C_IsEnabledIT_TX.....	300
2.16.53	LL_I2C_EnableIT_RX.....	300
2.16.54	LL_I2C_DisableIT_RX.....	301
2.16.55	LL_I2C_IsEnabledIT_RX.....	301
2.16.56	LL_I2C_EnableIT_ADDR.....	301
2.16.57	LL_I2C_DisableIT_ADDR.....	301
2.16.58	LL_I2C_IsEnabledIT_ADDR.....	302
2.16.59	LL_I2C_EnableIT_NACK.....	302
2.16.60	LL_I2C_DisableIT_NACK.....	302
2.16.61	LL_I2C_IsEnabledIT_NACK.....	303
2.16.62	LL_I2C_EnableIT_STOP.....	303
2.16.63	LL_I2C_DisableIT_STOP.....	303
2.16.64	LL_I2C_IsEnabledIT_STOP.....	303
2.16.65	LL_I2C_EnableIT_TC.....	304
2.16.66	LL_I2C_DisableIT_TC.....	304
2.16.67	LL_I2C_IsEnabledIT_TC.....	304
2.16.68	LL_I2C_EnableIT_ERR.....	304
2.16.69	LL_I2C_DisableIT_ERR.....	305
2.16.70	LL_I2C_IsEnabledIT_ERR.....	305
2.16.71	LL_I2C_EnableIT.....	305
2.16.72	LL_I2C_DisableIT.....	306
2.16.73	LL_I2C_IsEnabledIT.....	306
2.16.74	LL_I2C_IsActiveFlag_TXE.....	306
2.16.75	LL_I2C_IsActiveFlag_TXIS.....	306
2.16.76	LL_I2C_IsActiveFlag_RXNE.....	307
2.16.77	LL_I2C_IsActiveFlag_ADDR.....	307
2.16.78	LL_I2C_IsActiveFlag_NACK.....	307
2.16.79	LL_I2C_IsActiveFlag_STOP.....	307
2.16.80	LL_I2C_IsActiveFlag_TC.....	308
2.16.81	LL_I2C_IsActiveFlag_TCR.....	308
2.16.82	LL_I2C_IsActiveFlag_BERR.....	308

2.16.83	LL_I2C_IsActiveFlag_ARLO	309
2.16.84	LL_I2C_IsActiveFlag_OVR.....	309
2.16.85	LL_I2C_IsActiveFlag_TIMEOUT	309
2.16.86	LL_I2C_IsActiveFlag_BUSY	309
2.16.87	LL_I2C_IsActiveFlag	310
2.16.88	LL_I2C_ClearFlag_ADDR.....	310
2.16.89	LL_I2C_ClearFlag_TXIS.....	310
2.16.90	LL_I2C_ClearFlag_NACK.....	311
2.16.91	LL_I2C_ClearFlag_STOP	311
2.16.92	LL_I2C_ClearFlag_TXE	311
2.16.93	LL_I2C_ClearFlag_BERR	311
2.16.94	LL_I2C_ClearFlag_ARLO	312
2.16.95	LL_I2C_ClearFlag_OVR	312
2.16.96	LL_I2C_ClearFlag	312
2.16.97	LL_I2C_EnableAutoEndMode.....	313
2.16.98	LL_I2C_DisableAutoEndMode.....	313
2.16.99	LL_I2C_IsEnabledAutoEndMode.....	313
2.16.100	LL_I2C_EnableReloadMode	313
2.16.101	LL_I2C_DisableReloadMode.....	314
2.16.102	LL_I2C_IsEnabledReloadMode	314
2.16.103	LL_I2C_SetTransferSize	314
2.16.104	LL_I2C_GetTransferSize.....	315
2.16.105	LL_I2C_AcknowledgeNextData	315
2.16.106	LL_I2C_GenerateStartCondition.....	315
2.16.107	LL_I2C_IsGenerateStartCondition	315
2.16.108	LL_I2C_GenerateStopCondition	316
2.16.109	LL_I2C_IsGenerateStopCondition	316
2.16.110	LL_I2C_EnableAuto10BitRead	316
2.16.111	LL_I2C_DisableAuto10BitRead	317
2.16.112	LL_I2C_IsEnabledAuto10BitRead.....	317
2.16.113	LL_I2C_SetTransferRequest	317
2.16.114	LL_I2C_GetTransferRequest.....	317
2.16.115	LL_I2C_SetSlaveAddr	318
2.16.116	LL_I2C_GetSlaveAddr.....	318
2.16.117	LL_I2C_HandleTransfer.....	318
2.16.118	LL_I2C_GetTransferDirection	319
2.16.119	LL_I2C_GetAddressMatchCode	319
2.16.120	LL_I2C_ReceiveData8	319
2.16.121	LL_I2C_TransmitData8.....	320
2.16.122	LL_I2C_Init	320
2.16.123	LL_I2C_DeInit	320
2.16.124	LL_I2C_StructInit	321
2.17	LCDLED 模块.....	321
2.17.1	LL_LCDLED_SetMode	321
2.17.2	LL_LCDLED_GetMode.....	321

2.17.3	LL_LCDLED_SetDuty	322
2.17.4	LL_LCDLED_GetDuty	322
2.17.5	LL_LCDLED_SetBrightLevel	323
2.17.6	LL_LCDLED_GetBrightLevel	323
2.17.7	LL_LCDLED_LED_SetBrightMode	324
2.17.8	LL_LCDLED_LED_GetBrightMode	324
2.17.9	LL_LCDLED_EnableClockHold	324
2.17.10	LL_LCDLED_DisableClockHold	324
2.17.11	LL_LCDLED_IsEnabledClockHold	325
2.17.12	LL_LCDLED_SetClockDivision	325
2.17.13	LL_LCDLED_GetClockDivision	325
2.17.14	LL_LCDLED_LCD_SetClockSource	326
2.17.15	LL_LCDLED_LCD_GetClockSource	326
2.17.16	LL_LCDLED_Enable	326
2.17.17	LL_LCDLED_Disable	326
2.17.18	LL_LCDLED_IsEnabled	327
2.17.19	LL_LCDLED_SetRAMData	327
2.17.20	LL_LCDLED_GetRAMData	327
2.18	LPTIM 模块	328
2.18.1	LL_LPTIM_DeInit	328
2.18.2	LL_LPTIM_StructInit	328
2.18.3	LL_LPTIM_Init	329
2.18.4	LL_LPTIM_Enable	329
2.18.5	LL_LPTIM_Disable	329
2.18.6	LL_LPTIM_IsEnabled	330
2.18.7	LL_LPTIM_StartCounter	330
2.18.8	LL_LPTIM_EnableResetAfterRead	330
2.18.9	LL_LPTIM_DisableResetAfterRead	331
2.18.10	LL_LPTIM_IsEnabledResetAfterRead	331
2.18.11	LL_LPTIM_ResetCounter	331
2.18.12	LL_LPTIM_SetUpdateMode	331
2.18.13	LL_LPTIM_GetUpdateMode	332
2.18.14	LL_LPTIM_SetAutoReload	332
2.18.15	LL_LPTIM_GetAutoReload	332
2.18.16	LL_LPTIM_SetCompare	333
2.18.17	LL_LPTIM_GetCompare	333
2.18.18	LL_LPTIM_GetCounter	333
2.18.19	LL_LPTIM_SetCounterMode	333
2.18.20	LL_LPTIM_GetCounterMode	334
2.18.21	LL_LPTIM_ConfigOutput	334
2.18.22	LL_LPTIM_SetWaveform	335
2.18.23	LL_LPTIM_GetWaveform	335
2.18.24	LL_LPTIM_SetPolarity	335
2.18.25	LL_LPTIM_GetPolarity	336
2.18.26	LL_LPTIM_SetPrescaler	336

2.18.27	LL_LPTIM_GetPrescaler	336
2.18.28	LL_LPTIM_SetInput1Polarity	337
2.18.29	LL_LPTIM_GetInput1Polarity	337
2.18.30	LL_LPTIM_SetInput2Polarity	337
2.18.31	LL_LPTIM_GetInput2Polarity	338
2.18.32	LL_LPTIM_SetInput1LSI	338
2.18.33	LL_LPTIM_GetInput1LSI	338
2.18.34	LL_LPTIM_SetInput2LSI	339
2.18.35	LL_LPTIM_GetInput2LSI	339
2.18.36	LL_LPTIM_EnableTimeout	339
2.18.37	LL_LPTIM_DisableTimeout	340
2.18.38	LL_LPTIM_IsEnabledTimeout	340
2.18.39	LL_LPTIM_TrigSw	340
2.18.40	LL_LPTIM_ConfigTrigger	340
2.18.41	LL_LPTIM_SetTriggerFilter	341
2.18.42	LL_LPTIM_GetTriggerFilter	341
2.18.43	LL_LPTIM_GetTriggerSource	342
2.18.44	LL_LPTIM_GetTriggerPolarity	342
2.18.45	LL_LPTIM_SetClockSource	342
2.18.46	LL_LPTIM_GetClockSource	343
2.18.47	LL_LPTIM_ConfigClock	343
2.18.48	LL_LPTIM_SetClockFilter	343
2.18.49	LL_LPTIM_GetClockFilter	344
2.18.50	LL_LPTIM_GetClockPolarity	344
2.18.51	LL_LPTIM_GetFilterClockDiv	344
2.18.52	LL_LPTIM_SetEncoderMode	345
2.18.53	LL_LPTIM_GetEncoderMode	345
2.18.54	LL_LPTIM_EnableEncoderMode	345
2.18.55	LL_LPTIM_DisableEncoderMode	346
2.18.56	LL_LPTIM_IsEnabledEncoderMode	346
2.18.57	LL_LPTIM_SetDualEncoderMode	346
2.18.58	LL_LPTIM_GetDualEncoderMode	347
2.18.59	LL_LPTIM_EnableDualEncoderMode	347
2.18.60	LL_LPTIM_DisableDualEncoderMode	347
2.18.61	LL_LPTIM_IsEnabledDualEncoderMode	347
2.18.62	LL_LPTIM_ClearFLAG_CMPM	348
2.18.63	LL_LPTIM_IsActiveFlag_CMPM	348
2.18.64	LL_LPTIM_ClearFLAG_ARRM	348
2.18.65	LL_LPTIM_IsActiveFlag_ARRM	349
2.18.66	LL_LPTIM_ClearFLAG_EXTTRIG	349
2.18.67	LL_LPTIM_IsActiveFlag_EXTTRIG	349
2.18.68	LL_LPTIM_ClearFLAG_CMPOK	349
2.18.69	LL_LPTIM_IsActiveFlag_CMPOK	350
2.18.70	LL_LPTIM_ClearFLAG_ARROK	350
2.18.71	LL_LPTIM_IsActiveFlag_ARROK	350
2.18.72	LL_LPTIM_ClearFLAG_UP	351

2.18.73	LL_LPTIM_IsActiveFlag_UP	351
2.18.74	LL_LPTIM_ClearFLAG_DOWN	351
2.18.75	LL_LPTIM_IsActiveFlag_DOWN.....	351
2.18.76	LL_LPTIM_ClearFLAG_IN1IT.....	352
2.18.77	LL_LPTIM_IsActiveFlag_IN1IT	352
2.18.78	LL_LPTIM_ClearFLAG_IN2IT.....	352
2.18.79	LL_LPTIM_IsActiveFlag_IN2IT	352
2.18.80	LL_LPTIM_ClearFLAG_INB2B	353
2.18.81	LL_LPTIM_IsActiveFlag_INB2B	353
2.18.82	LL_LPTIM_ClearFLAG_DUALError	353
2.18.83	LL_LPTIM_IsActiveFlag_DUALError.....	354
2.18.84	LL_LPTIM_ClearFLAG.....	354
2.18.85	LL_LPTIM_IsActiveFlag	354
2.18.86	LL_LPTIM_EnableIT_CMPM.....	355
2.18.87	LL_LPTIM_DisableIT_CMPM.....	355
2.18.88	LL_LPTIM_IsEnabledIT_CMPM	355
2.18.89	LL_LPTIM_EnableIT_ARRM	356
2.18.90	LL_LPTIM_DisableIT_ARRM.....	356
2.18.91	LL_LPTIM_IsEnabledIT_ARRM	356
2.18.92	LL_LPTIM_EnableIT_EXTTRIG.....	356
2.18.93	LL_LPTIM_DisableIT_EXTTRIG.....	357
2.18.94	LL_LPTIM_IsEnabledIT_EXTTRIG.....	357
2.18.95	LL_LPTIM_EnableIT_CMPMOK	357
2.18.96	LL_LPTIM_DisableIT_CMPOK	357
2.18.97	LL_LPTIM_IsEnabledIT_CMPOK.....	358
2.18.98	LL_LPTIM_EnableIT_ARROK.....	358
2.18.99	LL_LPTIM_DisableIT_ARROK.....	358
2.18.100	LL_LPTIM_IsEnabledIT_ARROK	359
2.18.101	LL_LPTIM_EnableIT_UP.....	359
2.18.102	LL_LPTIM_DisableIT_UP.....	359
2.18.103	LL_LPTIM_IsEnabledIT_UP.....	359
2.18.104	LL_LPTIM_EnableIT_DOWN	360
2.18.105	LL_LPTIM_DisableIT_DOWN.....	360
2.18.106	LL_LPTIM_IsEnabledIT_DOWN.....	360
2.18.107	LL_LPTIM_EnableIT_IN1.....	360
2.18.108	LL_LPTIM_DisableIT_IN1	361
2.18.109	LL_LPTIM_IsEnabledIT_IN1.....	361
2.18.110	LL_LPTIM_EnableIT_IN2.....	361
2.18.111	LL_LPTIM_DisableIT_IN2	362
2.18.112	LL_LPTIM_IsEnabledIT_IN2.....	362
2.18.113	LL_LPTIM_EnableIT_INB2B	362
2.18.114	LL_LPTIM_DisableIT_INB2B	362
2.18.115	LL_LPTIM_IsEnabledIT_INB2B	363
2.18.116	LL_LPTIM_EnableIT_DUALError	363
2.18.117	LL_LPTIM_DisableIT_DUALError	363
2.18.118	LL_LPTIM_IsEnabledIT_DUALError	363

2.18.119	LL_LPTIM_EnableIT.....	364
2.18.120	LL_LPTIM_DisableIT.....	364
2.18.121	LL_LPTIM_IsEnabledIT.....	365
2.19	LPUART 模块.....	365
2.19.1	LL_LPUART_Enable.....	366
2.19.2	LL_LPUART_Disable.....	366
2.19.3	LL_LPUART_IsEnabled.....	366
2.19.4	LL_LPUART_EnableFIFO.....	367
2.19.5	LL_LPUART_DisableFIFO.....	367
2.19.6	LL_LPUART_IsEnabledFIFO.....	367
2.19.7	LL_LPUART_SetTXFIFOThreshold.....	368
2.19.8	LL_LPUART_GetTXFIFOThreshold.....	368
2.19.9	LL_LPUART_SetRXFIFOThreshold.....	368
2.19.10	LL_LPUART_GetRXFIFOThreshold.....	369
2.19.11	LL_LPUART_ConfigFIFOsThreshold.....	369
2.19.12	LL_LPUART_EnableInStopMode.....	370
2.19.13	LL_LPUART_DisableInStopMode.....	370
2.19.14	LL_LPUART_IsEnabledInStopMode.....	370
2.19.15	LL_LPUART_EnableDirectionRx.....	371
2.19.16	LL_LPUART_DisableDirectionRx.....	371
2.19.17	LL_LPUART_EnableDirectionTx.....	371
2.19.18	LL_LPUART_DisableDirectionTx.....	371
2.19.19	LL_LPUART_SetTransferDirection.....	372
2.19.20	LL_LPUART_GetTransferDirection.....	372
2.19.21	LL_LPUART_SetParity.....	372
2.19.22	LL_LPUART_GetParity.....	373
2.19.23	LL_LPUART_SetWakeUpMethod.....	373
2.19.24	LL_LPUART_GetWakeUpMethod.....	373
2.19.25	LL_LPUART_SetDataWidth.....	374
2.19.26	LL_LPUART_GetDataWidth.....	374
2.19.27	LL_LPUART_EnableMuteMode.....	374
2.19.28	LL_LPUART_DisableMuteMode.....	375
2.19.29	LL_LPUART_IsEnabledMuteMode.....	375
2.19.30	LL_LPUART_SetPrescaler.....	375
2.19.31	LL_LPUART_GetPrescaler.....	376
2.19.32	LL_LPUART_SetStopBitsLength.....	376
2.19.33	LL_LPUART_GetStopBitsLength.....	377
2.19.34	LL_LPUART_ConfigCharacter.....	377
2.19.35	LL_LPUART_SetTXRXSwap.....	377
2.19.36	LL_LPUART_GetTXRXSwap.....	378
2.19.37	LL_LPUART_SetRXPInLevel.....	378
2.19.38	LL_LPUART_GetRXPInLevel.....	378
2.19.39	LL_LPUART_SetTXPinLevel.....	379
2.19.40	LL_LPUART_GetTXPinLevel.....	379
2.19.41	LL_LPUART_SetBinaryDataLogic.....	379

2.19.42	LL_LPUART_GetBinaryDataLogic	380
2.19.43	LL_LPUART_SetTransferBitOrder	380
2.19.44	LL_LPUART_GetTransferBitOrder	380
2.19.45	LL_LPUART_ConfigNodeAddress	381
2.19.46	LL_LPUART_GetNodeAddress	381
2.19.47	LL_LPUART_GetNodeAddressLen	381
2.19.48	LL_LPUART_EnableRTSHWFlowCtrl	382
2.19.49	LL_LPUART_DisableRTSHWFlowCtrl	382
2.19.50	LL_LPUART_EnableCTSHWFlowCtrl	382
2.19.51	LL_LPUART_DisableCTSHWFlowCtrl	382
2.19.52	LL_LPUART_SetHWFlowCtrl	383
2.19.53	LL_LPUART_GetHWFlowCtrl	383
2.19.54	LL_LPUART_EnableOverrunDetect	383
2.19.55	LL_LPUART_DisableOverrunDetect	384
2.19.56	LL_LPUART_IsEnabledOverrunDetect	384
2.19.57	LL_LPUART_SetWKUPType	384
2.19.58	LL_LPUART_GetWKUPType	385
2.19.59	LL_LPUART_SetBaudRate	385
2.19.60	LL_LPUART_GetBaudRate	386
2.19.61	LL_LPUART_EnableHalfDuplex	386
2.19.62	LL_LPUART_DisableHalfDuplex	386
2.19.63	LL_LPUART_IsEnabledHalfDuplex	387
2.19.64	LL_LPUART_SetDEDeassertionTime	387
2.19.65	LL_LPUART_GetDEDeassertionTime	387
2.19.66	LL_LPUART_SetDEAssertionTime	388
2.19.67	LL_LPUART_GetDEAssertionTime	388
2.19.68	LL_LPUART_EnableDEMode	388
2.19.69	LL_LPUART_DisableDEMode	388
2.19.70	LL_LPUART_IsEnabledDEMode	389
2.19.71	LL_LPUART_SetDESignalPolarity	389
2.19.72	LL_LPUART_GetDESignalPolarity	389
2.19.73	LL_LPUART_IsActiveFlag_PE	390
2.19.74	LL_LPUART_IsActiveFlag_FE	390
2.19.75	LL_LPUART_IsActiveFlag_NE	390
2.19.76	LL_LPUART_IsActiveFlag_ORE	391
2.19.77	LL_LPUART_IsActiveFlag_IDLE	391
2.19.78	LL_LPUART_IsActiveFlag_RXNE_RXFNE	391
2.19.79	LL_LPUART_IsActiveFlag_TC	391
2.19.80	LL_LPUART_IsActiveFlag_TXE_TXFNF	392
2.19.81	LL_LPUART_IsActiveFlag_nCTS	392
2.19.82	LL_LPUART_IsActiveFlag_CTS	392
2.19.83	LL_LPUART_IsActiveFlag_BUSY	393
2.19.84	LL_LPUART_IsActiveFlag_CM	393
2.19.85	LL_LPUART_IsActiveFlag_SBK	393
2.19.86	LL_LPUART_IsActiveFlag_RWU	393
2.19.87	LL_LPUART_IsActiveFlag_WKUP	394

2.19.88	LL_LPUART_IsActiveFlag_TEACK	394
2.19.89	LL_LPUART_IsActiveFlag_REACK	394
2.19.90	LL_LPUART_IsActiveFlag_TXFE	395
2.19.91	LL_LPUART_IsActiveFlag_RXFE	395
2.19.92	LL_LPUART_IsActiveFlag_TXFT	395
2.19.93	LL_LPUART_IsActiveFlag_RXFT	396
2.19.94	LL_LPUART_ClearFlag_PE	396
2.19.95	LL_LPUART_ClearFlag_FE	396
2.19.96	LL_LPUART_ClearFlag_NE	396
2.19.97	LL_LPUART_ClearFlag_ORE	397
2.19.98	LL_LPUART_ClearFlag_IDLE	397
2.19.99	LL_LPUART_ClearFlag_TC	397
2.19.100	LL_LPUART_ClearFlag_nCTS	397
2.19.101	LL_LPUART_ClearFlag_CM	398
2.19.102	LL_LPUART_ClearFlag_WKUP	398
2.19.103	LL_LPUART_EnableIT_IDLE	398
2.19.104	LL_LPUART_EnableIT_RXNE_RXFNE	399
2.19.105	LL_LPUART_EnableIT_TC	399
2.19.106	LL_LPUART_EnableIT_TXE_TXFNF	399
2.19.107	LL_LPUART_EnableIT_PE	399
2.19.108	LL_LPUART_EnableIT_CM	400
2.19.109	LL_LPUART_EnableIT_TXFE	400
2.19.110	LL_LPUART_EnableIT_RXFF	400
2.19.111	LL_LPUART_EnableIT_ERROR	400
2.19.112	LL_LPUART_EnableIT_CTS	401
2.19.113	LL_LPUART_EnableIT_WKUP	401
2.19.114	LL_LPUART_EnableIT_TXFT	401
2.19.115	LL_LPUART_EnableIT_RXFT	402
2.19.116	LL_LPUART_DisableIT_IDLE	402
2.19.117	LL_LPUART_DisableIT_RXNE_RXFNE	402
2.19.118	LL_LPUART_DisableIT_TC	402
2.19.119	LL_LPUART_DisableIT_TXE_TXFNF	403
2.19.120	LL_LPUART_DisableIT_PE	403
2.19.121	LL_LPUART_DisableIT_CM	403
2.19.122	LL_LPUART_DisableIT_TXFE	403
2.19.123	LL_LPUART_DisableIT_RXFF	404
2.19.124	LL_LPUART_DisableIT_ERROR	404
2.19.125	LL_LPUART_DisableIT_CTS	404
2.19.126	LL_LPUART_DisableIT_WKUP	405
2.19.127	LL_LPUART_DisableIT_TXFT	405
2.19.128	LL_LPUART_DisableIT_RXFT	405
2.19.129	LL_LPUART_IsEnabledIT_IDLE	405
2.19.130	LL_LPUART_IsEnabledIT_RXNE_RXFNE	406
2.19.131	LL_LPUART_IsEnabledIT_TC	406
2.19.132	LL_LPUART_IsEnabledIT_TXE_TXFNF	406
2.19.133	LL_LPUART_IsEnabledIT_PE	407

2.19.134	LL_LPUART_IsEnabledIT_CM	407
2.19.135	LL_LPUART_IsEnabledIT_TXFE	407
2.19.136	LL_LPUART_IsEnabledIT_RXFF	407
2.19.137	LL_LPUART_IsEnabledIT_ERROR	408
2.19.138	LL_LPUART_IsEnabledIT_CTS	408
2.19.139	LL_LPUART_IsEnabledIT_WKUP	408
2.19.140	LL_LPUART_IsEnabledIT_TXFT	409
2.19.141	LL_LPUART_IsEnabledIT_RXFT	409
2.19.142	LL_LPUART_EnableDMAReq_RX	409
2.19.143	LL_LPUART_DisableDMAReq_RX	409
2.19.144	LL_LPUART_IsEnabledDMAReq_RX	410
2.19.145	LL_LPUART_EnableDMAReq_TX	410
2.19.146	LL_LPUART_DisableDMAReq_TX	410
2.19.147	LL_LPUART_IsEnabledDMAReq_TX	411
2.19.148	LL_LPUART_EnableDMADeactOnRxErr	411
2.19.149	LL_LPUART_DisableDMADeactOnRxErr	411
2.19.150	LL_LPUART_IsEnabledDMADeactOnRxErr	411
2.19.151	LL_LPUART_DMA_GetRegAddr	412
2.19.152	LL_LPUART_ReceiveData8	412
2.19.153	LL_LPUART_ReceiveData9	412
2.19.154	LL_LPUART_TransmitData8	413
2.19.155	LL_LPUART_TransmitData9	413
2.19.156	LL_LPUART_RequestBreakSending	413
2.19.157	LL_LPUART_RequestEnterMuteMode	414
2.19.158	LL_LPUART_RequestRxDataFlush	414
2.19.159	LL_LPUART_RequestTxDataFlush	414
2.19.160	LL_LPUART_DeInit	414
2.19.161	LL_LPUART_Init	415
2.19.162	LL_LPUART_StructInit	415
2.20	OPAMP 模块	415
2.20.1	LL_OPAMP_SetInputNonInverting	416
2.20.2	LL_OPAMP_GetInputNonInverting	416
2.20.3	LL_OPAMP_SetInputInverting	416
2.20.4	LL_OPAMP_GetInputInverting	417
2.20.5	LL_OPAMP_SetPGAGain	417
2.20.6	LL_OPAMP_GetPGAGain	418
2.20.7	LL_OPAMP_SetFunctionalMode	418
2.20.8	LL_OPAMP_GetFunctionalMode	418
2.20.9	LL_OPAMP_SetTrimmingSource	419
2.20.10	LL_OPAMP_GetTrimmingSource	419
2.20.11	LL_OPAMP_SetOutputMode	419
2.20.12	LL_OPAMP_GetOutputMode	420
2.20.13	LL_OPAMP_Enable	420
2.20.14	LL_OPAMP_Disable	420
2.20.15	LL_OPAMP_IsEnabled	421

2.20.16	LL_OPAMP_Lock.....	421
2.20.17	LL_OPAMP_IsLocked.....	421
2.20.18	LL_OPAMP_DeInit.....	421
2.20.19	LL_OPAMP_Init.....	422
2.20.20	LL_OPAMP_StructInit.....	422
2.21	PLA 模块.....	422
2.21.1	LL_PLA_SetMUXAInput.....	423
2.21.2	LL_PLA_GetMUXAInput.....	423
2.21.3	LL_PLA_SetMUXBInput.....	424
2.21.4	LL_PLA_GetMUXBInput.....	424
2.21.5	LL_PLA_EnableOutput.....	425
2.21.6	LL_PLA_DisableOutput.....	425
2.21.7	LL_PLA_IsEnabledOutput.....	425
2.21.8	LL_PLA_SetFunctionMode.....	426
2.21.9	LL_PLA_GetFunctionMode.....	426
2.21.10	LL_PLA_SetOutputMode.....	426
2.21.11	LL_PLA_GetOutputMode.....	427
2.21.12	LL_PLA_DFF_Reset.....	427
2.21.13	LL_PLA_DFF_SetClockInverseMode.....	427
2.21.14	LL_PLA_DFF_GetClockInverseMode.....	428
2.21.15	LL_PLA_DFF_SetClockSource.....	428
2.21.16	LL_PLA_DFF_GetClockSource.....	429
2.21.17	LL_PLA_EnableIT_Rising.....	429
2.21.18	LL_PLA_DisableIT_Rising.....	429
2.21.19	LL_PLA_IsEnabledIT_Rising.....	430
2.21.20	LL_PLA_EnableIT_Falling.....	430
2.21.21	LL_PLA_DisableIT_Falling.....	430
2.21.22	LL_PLA_IsEnabledIT_Falling.....	431
2.21.23	LL_PLA_IsActiveFlag_Rising.....	431
2.21.24	LL_PLA_IsActiveFlag_Falling.....	431
2.21.25	LL_PLA_ClearFlag_Rising.....	432
2.21.26	LL_PLA_ClearFlag_Falling.....	432
2.21.27	LL_PLA_Enable.....	432
2.21.28	LL_PLA_Disable.....	433
2.21.29	LL_PLA_IsEnabled.....	433
2.21.30	LL_PLA_ReadOutputLevel.....	433
2.22	RCC 模块.....	434
2.22.1	LL_RCC_HSE_EnableBypass.....	434
2.22.2	LL_RCC_HSE_DisableBypass.....	434
2.22.3	LL_RCC_HSE_IsEnabledBypass.....	435
2.22.4	LL_RCC_HSE_Enable.....	435
2.22.5	LL_RCC_HSE_Disable.....	435
2.22.6	LL_RCC_HSE_IsEnabled.....	435
2.22.7	LL_RCC_HSE_SetState.....	436
2.22.8	LL_RCC_HSE_GetState.....	436

2.22.9	LL_RCC_HSE_SetStableCycle.....	436
2.22.10	LL_RCC_HSE_GetStableCycle.....	437
2.22.11	LL_RCC_HSE_SetDivision.....	437
2.22.12	LL_RCC_HSE_GetDivision.....	438
2.22.13	LL_RCC_HSE_SetOutPin.....	438
2.22.14	LL_RCC_HSE_GetOutPin.....	438
2.22.15	LL_RCC_HSE_SetInPin.....	438
2.22.16	LL_RCC_HSE_GetInPin.....	439
2.22.17	LL_RCC_HSE_SetDrive.....	439
2.22.18	LL_RCC_HSE_GetDrive.....	439
2.22.19	LL_RCC_HSE_IsReady.....	440
2.22.20	LL_RCC_LSE_EnableCSS.....	440
2.22.21	LL_RCC_LSE_DisableCSS.....	440
2.22.22	LL_RCC_LSE_IsEnabledCSS.....	440
2.22.23	LL_RCC_LSE_EnableBypass.....	441
2.22.24	LL_RCC_LSE_DisableBypass.....	441
2.22.25	LL_RCC_LSE_IsEnabledBypass.....	441
2.22.26	LL_RCC_LSE_Enable.....	441
2.22.27	LL_RCC_LSE_Disable.....	442
2.22.28	LL_RCC_LSE_IsEnabled.....	442
2.22.29	LL_RCC_LSE_SetState.....	442
2.22.30	LL_RCC_LSE_GetState.....	443
2.22.31	LL_RCC_LSE_SetDriveCapability.....	443
2.22.32	LL_RCC_LSE_GetDriveCapability.....	443
2.22.33	LL_RCC_LSE_SetStableCycle.....	443
2.22.34	LL_RCC_LSE_GetStableCycle.....	444
2.22.35	LL_RCC_LSE_IsReady.....	444
2.22.36	LL_RCC_LSE_IsCSSDetected.....	445
2.22.37	LL_RCC_HSI_EnableAlwaysON.....	445
2.22.38	LL_RCC_HSI_DisableAlwaysON.....	445
2.22.39	LL_RCC_HSI_IsEnabledAlwaysON.....	445
2.22.40	LL_RCC_HSI_Enable.....	446
2.22.41	LL_RCC_HSI_Disable.....	446
2.22.42	LL_RCC_HSI_IsEnabled.....	446
2.22.43	LL_RCC_HSI_SetState.....	446
2.22.44	LL_RCC_HSI_GetState.....	447
2.22.45	LL_RCC_HSI_IsReady.....	447
2.22.46	LL_RCC_HSI_SetCalibTrimming.....	447
2.22.47	LL_RCC_HSI_GetCalibTrimming.....	448
2.22.48	LL_RCC_HSI_SetDivision.....	448
2.22.49	LL_RCC_HSI_GetDivision.....	448
2.22.50	LL_RCC_HSI_SetStableCycle.....	448
2.22.51	LL_RCC_HSI_GetStableCycle.....	449
2.22.52	LL_RCC_LSI_Enable.....	449
2.22.53	LL_RCC_LSI_Disable.....	449
2.22.54	LL_RCC_LSI_IsEnabled.....	450

2.22.55	LL_RCC_LSI_SetState.....	450
2.22.56	LL_RCC_LSI_GetState.....	450
2.22.57	LL_RCC_LSI_IsReady.....	450
2.22.58	LL_RCC_LSI_SetCalibTrimming.....	451
2.22.59	LL_RCC_LSI_GetCalibTrimming.....	451
2.22.60	LL_RCC_MSI_Enable.....	451
2.22.61	LL_RCC_MSI_Disable.....	452
2.22.62	LL_RCC_MSI_IsEnabled.....	452
2.22.63	LL_RCC_MSI_SetState.....	452
2.22.64	LL_RCC_MSI_GetState.....	452
2.22.65	LL_RCC_MSI_IsReady.....	453
2.22.66	LL_RCC_MSI_SetCalibTrimming.....	453
2.22.67	LL_RCC_MSI_GetCalibTrimming.....	453
2.22.68	LL_RCC_MSI_SetStableCycle.....	453
2.22.69	LL_RCC_MSI_GetStableCycle.....	454
2.22.70	LL_RCC_LSCO_Enable.....	454
2.22.71	LL_RCC_LSCO_Disable.....	454
2.22.72	LL_RCC_LSCO_IsEnabled.....	455
2.22.73	LL_RCC_LSCO_SetSource.....	455
2.22.74	LL_RCC_LSCO_GetSource.....	455
2.22.75	LL_RCC_PLL_EnableCSS.....	456
2.22.76	LL_RCC_PLL_IsEnabledCSS.....	456
2.22.77	LL_RCC_PLL_Enable.....	456
2.22.78	LL_RCC_PLL_Disable.....	456
2.22.79	LL_RCC_PLL_IsEnabled.....	457
2.22.80	LL_RCC_PLL_IsReady.....	457
2.22.81	LL_RCC_PLL_EnableDomain_SYS.....	457
2.22.82	LL_RCC_PLL_DisableDomain_SYS.....	457
2.22.83	LL_RCC_PLL_IsEnabledDomain_SYS.....	458
2.22.84	LL_RCC_PLL_EnableDomain_TIM1.....	458
2.22.85	LL_RCC_PLL_DisableDomain_TIM1.....	458
2.22.86	LL_RCC_PLL_IsEnabledDomain_TIM1.....	459
2.22.87	LL_RCC_PLL_ConfigDomain_SYS.....	459
2.22.88	LL_RCC_PLL_ConfigDomain_TIM1.....	459
2.22.89	LL_RCC_PLL_GetM.....	460
2.22.90	LL_RCC_PLL_SetM.....	460
2.22.91	LL_RCC_PLL_GetR.....	461
2.22.92	LL_RCC_PLL_SetR.....	461
2.22.93	LL_RCC_PLL_GetQ.....	461
2.22.94	LL_RCC_PLL_SetQ.....	462
2.22.95	LL_RCC_PLL_GetN.....	462
2.22.96	LL_RCC_PLL_SetN.....	462
2.22.97	LL_RCC_PLL_GetMainSource.....	463
2.22.98	LL_RCC_PLL_SetMainSource.....	463
2.22.99	LL_RCC_SetSysClkSource.....	463
2.22.100	LL_RCC_GetSysClkSource.....	464

2.22.101	LL_RCC_SetAHBPrescaler	464
2.22.102	LL_RCC_GetAHBPrescaler.....	464
2.22.103	LL_RCC_SetAPB1Prescaler	465
2.22.104	LL_RCC_GetAPB1Prescaler	465
2.22.105	LL_RCC_SetAPB2Prescaler	465
2.22.106	LL_RCC_GetAPB2Prescaler	466
2.22.107	LL_RCC_ConfigMCO	466
2.22.108	LL_RCC_GetMCOPrescaler	466
2.22.109	LL_RCC_GetMCOSource.....	467
2.22.110	LL_RCC_SetUSARTClockSource	467
2.22.111	LL_RCC_GetUSARTClockSource	467
2.22.112	LL_RCC_SetLPUARTClockSource.....	468
2.22.113	LL_RCC_GetLPUARTClockSource	468
2.22.114	LL_RCC_SetI2CClockSource	469
2.22.115	LL_RCC_GetI2CClockSource	469
2.22.116	LL_RCC_SetLPTIMClockSource	469
2.22.117	LL_RCC_GetLPTIMClockSource	470
2.22.118	LL_RCC_SetTIMClockSource.....	470
2.22.119	LL_RCC_GetTIMClockSource	470
2.22.120	LL_RCC_SetADCClockSource	471
2.22.121	LL_RCC_GetADCClockSource.....	471
2.22.122	LL_RCC_RTC_SetClockSource	472
2.22.123	LL_RCC_RTC_GetClockSource.....	472
2.22.124	LL_RCC_RTC_SetFreqAdjust	472
2.22.125	LL_RCC_RTC_GetFreqAdjust	473
2.22.126	LL_RCC_RTC_Enable	473
2.22.127	LL_RCC_RTC_Disable.....	473
2.22.128	LL_RCC_RTC_IsEnabled	474
2.22.129	LL_RCC_RTC_ForceReset	474
2.22.130	LL_RCC_RTC_ReleaseReset.....	474
2.22.131	LL_RCC_RTC_EnableLowpowerMode	475
2.22.132	LL_RCC_RTC_DisableLowpowerMode	475
2.22.133	LL_RCC_RTC_IsEnabledLowpowerMode.....	475
2.22.134	LL_RCC_IsActiveRSTFlag_WWDG	475
2.22.135	LL_RCC_IsActiveRSTFlag_IWDG.....	476
2.22.136	LL_RCC_IsActiveRSTFlag_SFT	476
2.22.137	LL_RCC_IsActiveRSTFlag_SFT	476
2.22.138	LL_RCC_IsActiveRSTFlag_POR.....	476
2.22.139	LL_RCC_IsActiveRSTFlag_NRST.....	477
2.22.140	LL_RCC_IsActiveRSTFlag_OBL.....	477
2.22.141	LL_RCC_IsActiveRSTFlag_LOCKUP.....	477
2.22.142	LL_RCC_IsActiveRSTFlag_BOR	478
2.22.143	LL_RCC_IsActiveRSTFlag.....	478
2.22.144	LL_RCC_ClearRSTFlag.....	478
2.22.145	LL_RCC_EnableLOCKUPRST	478
2.22.146	LL_RCC_DisableLOCKUPRST	479

2.22.147	LL_RCC_IsEnabledLOCKUPRST	479
2.22.148	LL_RCC_ClearFlag_LSIRDY	479
2.22.149	LL_RCC_ClearFlag_LSERDY	480
2.22.150	LL_RCC_ClearFlag_MSIRDY	480
2.22.151	LL_RCC_ClearFlag_HSIRDY	480
2.22.152	LL_RCC_ClearFlag_HSERDY	480
2.22.153	LL_RCC_ClearFlag_PLLRDY	481
2.22.154	LL_RCC_ClearFlag_LSECSS.....	481
2.22.155	LL_RCC_ClearFlag_PLLCSS.....	481
2.22.156	LL_RCC_ClearFlag.....	481
2.22.157	LL_RCC_IsActiveFlag_LSIRDY.....	482
2.22.158	LL_RCC_IsActiveFlag_LSERDY	482
2.22.159	LL_RCC_IsActiveFlag_MSIRDY	482
2.22.160	LL_RCC_IsActiveFlag_HSIRDY	483
2.22.161	LL_RCC_IsActiveFlag_HSERDY	483
2.22.162	LL_RCC_IsActiveFlag_PLLRDY	483
2.22.163	LL_RCC_IsActiveFlag_LSECSS.....	483
2.22.164	LL_RCC_IsActiveFlag_PLLCSS.....	484
2.22.165	LL_RCC_IsActiveFlag.....	484
2.22.166	LL_RCC_EnableIT_LSIRDY	484
2.22.167	LL_RCC_DisableIT_LSIRDY	485
2.22.168	LL_RCC_IsEnabledIT_LSIRDY	485
2.22.169	LL_RCC_EnableIT_LSERDY	485
2.22.170	LL_RCC_DisableIT_LSERDY	485
2.22.171	LL_RCC_IsEnabledIT_LSERDY	486
2.22.172	LL_RCC_EnableIT_MSIRDY	486
2.22.173	LL_RCC_DisableIT_MSIRDY	486
2.22.174	LL_RCC_IsEnabledIT_MSIRDY	487
2.22.175	LL_RCC_EnableIT_HSIRDY.....	487
2.22.176	LL_RCC_DisableIT_HSIRDY.....	487
2.22.177	LL_RCC_IsEnabledIT_HSIRDY.....	487
2.22.178	LL_RCC_EnableIT_HSERDY.....	488
2.22.179	LL_RCC_DisableIT_HSERDY.....	488
2.22.180	LL_RCC_IsEnabledIT_HSERDY.....	488
2.22.181	LL_RCC_EnableIT_PLLRDY	488
2.22.182	LL_RCC_DisableIT_PLLRDY	489
2.22.183	LL_RCC_IsEnabledIT_PLLRDY	489
2.22.184	LL_RCC_EnableIT.....	489
2.22.185	LL_RCC_DisableIT.....	490
2.22.186	LL_RCC_IsEnabledIT.....	490
2.22.187	LL_RCC_BGR_SetCalibTrimming.....	490
2.22.188	LL_RCC_BGR_GetCalibTrimming.....	491
2.22.189	LL_RCC_DAC_SetCalibTrimming	491
2.22.190	LL_RCC_DAC_GetCalibTrimming.....	491
2.22.191	LL_RCC_OPA_SetCalibTrimming	491
2.22.192	LL_RCC_OPA_GetCalibTrimming.....	492

2.22.193	LL_RCC_DeInit	492
2.22.194	LL_RCC_GetSystemClocksFreq.....	492
2.22.195	LL_RCC_GetUSARTClockFreq	493
2.22.196	LL_RCC_GetI2CClockFreq	493
2.22.197	LL_RCC_GetLPUARTClockFreq.....	493
2.22.198	LL_RCC_GetLPTIMClockFreq.....	494
2.22.199	LL_RCC_GetADCClockFreq.....	494
2.22.200	LL_RCC_GetTIMClockFreq.....	494
2.22.201	LL_RCC_GetRTCClockFreq	494
2.23	SPI 模块.....	495
2.23.1	LL_SPI_Enable.....	495
2.23.2	LL_SPI_Disable.....	496
2.23.3	LL_SPI_IsEnabled.....	496
2.23.4	LL_SPI_SetMode	496
2.23.5	LL_SPI_GetMode.....	496
2.23.6	LL_SPI_SetClockPhase.....	497
2.23.7	LL_SPI_GetClockPhase	497
2.23.8	LL_SPI_SetClockPolarity	497
2.23.9	LL_SPI_GetClockPolarity.....	498
2.23.10	LL_SPI_SetBaudRatePrescaler	498
2.23.11	LL_SPI_GetBaudRatePrescaler.....	498
2.23.12	LL_SPI_SetTransferBitOrder	499
2.23.13	LL_SPI_GetTransferBitOrder.....	499
2.23.14	LL_SPI_SetTxFIFOThreshold	499
2.23.15	LL_SPI_GetTxFIFOThreshold.....	500
2.23.16	LL_SPI_SetRxFIFOThreshold	500
2.23.17	LL_SPI_GetRxFIFOThreshold.....	500
2.23.18	LL_SPI_SetNSSMode	501
2.23.19	LL_SPI_GetNSSMode	501
2.23.20	LL_SPI_SetNSSOutputLevel	501
2.23.21	LL_SPI_GetNSSOutputLevel.....	502
2.23.22	LL_SPI_SetSlaveNSSMode	502
2.23.23	LL_SPI_GetSlaveNSSMode.....	502
2.23.24	LL_SPI_IsActiveFlag_TXFT	502
2.23.25	LL_SPI_IsActiveFlag_TXFNF.....	503
2.23.26	LL_SPI_IsActiveFlag_TXFE	503
2.23.27	LL_SPI_IsActiveFlag_RXFT	503
2.23.28	LL_SPI_IsActiveFlag_RXFNE	503
2.23.29	LL_SPI_IsActiveFlag_RXFF	504
2.23.30	LL_SPI_IsActiveFlag_MMF.....	504
2.23.31	LL_SPI_IsActiveFlag_SME.....	504
2.23.32	LL_SPI_IsActiveFlag_OVR.....	505
2.23.33	LL_SPI_IsActiveFlag_BSY	505
2.23.34	LL_SPI_IsActiveFlag_UDR.....	505
2.23.35	LL_SPI_IsActiveFlag	505

2.23.36	LL_SPI_ClearFlag_MMF.....	506
2.23.37	LL_SPI_ClearFlag_SME.....	506
2.23.38	LL_SPI_ClearFlag_OVR.....	506
2.23.39	LL_SPI_ClearFlag.....	507
2.23.40	LL_SPI_ClearFlag_UDR.....	507
2.23.41	LL_SPI_EnableIT_MMF.....	507
2.23.42	LL_SPI_EnableIT_OVR.....	507
2.23.43	LL_SPI_EnableIT_UDR.....	508
2.23.44	LL_SPI_EnableIT_SME.....	508
2.23.45	LL_SPI_EnableIT_RXFF.....	508
2.23.46	LL_SPI_EnableIT_RXNE.....	509
2.23.47	LL_SPI_EnableIT_RXFT.....	509
2.23.48	LL_SPI_EnableIT_TXFT.....	509
2.23.49	LL_SPI_EnableIT_TXFNF.....	509
2.23.50	LL_SPI_EnableIT_TXFE.....	510
2.23.51	LL_SPI_EnableIT.....	510
2.23.52	LL_SPI_DisableIT_MMF.....	510
2.23.53	LL_SPI_DisableIT_OVR.....	511
2.23.54	LL_SPI_DisableIT_UDR.....	511
2.23.55	LL_SPI_DisableIT_SME.....	511
2.23.56	LL_SPI_DisableIT_RXFF.....	511
2.23.57	LL_SPI_DisableIT_RXNE.....	512
2.23.58	LL_SPI_DisableIT_RXFT.....	512
2.23.59	LL_SPI_DisableIT_TXFT.....	512
2.23.60	LL_SPI_DisableIT_TXFNF.....	512
2.23.61	LL_SPI_DisableIT_TXFE.....	513
2.23.62	LL_SPI_DisableIT.....	513
2.23.63	LL_SPI_IsEnabledIT_MMF.....	513
2.23.64	LL_SPI_IsEnabledIT_OVR.....	514
2.23.65	LL_SPI_IsEnabledIT_UDR.....	514
2.23.66	LL_SPI_IsEnabledIT_SME.....	514
2.23.67	LL_SPI_IsEnabledIT_RXFF.....	514
2.23.68	LL_SPI_IsEnabledIT_RXFNE.....	515
2.23.69	LL_SPI_IsEnabledIT_RXFT.....	515
2.23.70	LL_SPI_IsEnabledIT_TXFE.....	515
2.23.71	LL_SPI_IsEnabledIT_TXFNF.....	516
2.23.72	LL_SPI_IsEnabledIT_TXFT.....	516
2.23.73	LL_SPI_IsEnabledIT.....	516
2.23.74	LL_SPI_EnableDMAReq_RX.....	516
2.23.75	LL_SPI_DisableDMAReq_RX.....	517
2.23.76	LL_SPI_IsEnabledDMAReq_RX.....	517
2.23.77	LL_SPI_EnableDMAReq_TX.....	517
2.23.78	LL_SPI_DisableDMAReq_TX.....	518
2.23.79	LL_SPI_IsEnabledDMAReq_TX.....	518
2.23.80	LL_SPI_DMA_GetRegAddr.....	518
2.23.81	LL_SPI_ClearTransformFIFOData.....	518

2.23.82	LL_SPI_ClearReceiveFIFOData	519
2.23.83	LL_SPI_ReceiveData8	519
2.23.84	LL_SPI_TransmitData8	519
2.23.85	LL_SPI_DeInit.....	519
2.23.86	LL_SPI_Init	520
2.23.87	LL_SPI_StructInit.....	520
2.24	SYSTEM 模块.....	520
2.24.1	LL_SYSCFG_SetRemapMemory	520
2.24.2	LL_SYSCFG_GetRemapMemory.....	521
2.24.3	LL_SYSCFG_SetIRModEnvelopeSignal.....	521
2.24.4	LL_SYSCFG_GetIRModEnvelopeSignal	521
2.24.5	LL_SYSCFG_SetIRPolarity.....	522
2.24.6	LL_SYSCFG_GetIRPolarity	522
2.24.7	LL_SYSCFG_SetTIMBreakInputs.....	522
2.24.8	LL_SYSCFG_GetTIMBreakInputs	522
2.24.9	LL_DBGMCU_WriteData.....	523
2.24.10	LL_DBGMCU_ReadData	523
2.24.11	LL_DBGMCU_EnableDBGStopMode.....	523
2.24.12	LL_DBGMCU_DisableDBGStopMode.....	524
2.24.13	LL_DBGMCU_IsEnabledDBGStopMode	524
2.24.14	LL_DBGMCU_APB1_GRP1_FreezePeriph.....	524
2.24.15	LL_DBGMCU_APB1_GRP1_UnFreezePeriph.....	525
2.24.16	LL_DBGMCU_APB1_GRP1_IsFreezedPeriph.....	525
2.24.17	LL_DBGMCU_APB2_GRP1_FreezePeriph.....	526
2.24.18	LL_DBGMCU_APB2_GRP1_UnFreezePeriph.....	526
2.24.19	LL_DBGMCU_APB2_GRP1_IsFreezedPeriph.....	526
2.25	TIM 模块.....	527
2.25.1	LL_TIM_EnableCounter	529
2.25.2	LL_TIM_DisableCounter	530
2.25.3	LL_TIM_IsEnabledCounter	530
2.25.4	LL_TIM_EnableUpdateEvent	530
2.25.5	LL_TIM_DisableUpdateEvent	530
2.25.6	LL_TIM_IsEnabledUpdateEvent	531
2.25.7	LL_TIM_SetUpdateSource.....	531
2.25.8	LL_TIM_GetUpdateSource.....	531
2.25.9	LL_TIM_SetOnePulseMode.....	532
2.25.10	LL_TIM_GetOnePulseMode.....	532
2.25.11	LL_TIM_SetCounterMode.....	532
2.25.12	LL_TIM_GetCounterMode	533
2.25.13	LL_TIM_EnableARRPreload.....	533
2.25.14	LL_TIM_DisableARRPreload.....	533
2.25.15	LL_TIM_IsEnabledARRPreload.....	534
2.25.16	LL_TIM_SetClockDivision.....	534
2.25.17	LL_TIM_GetClockDivision	534
2.25.18	LL_TIM_SetCounter	535

2.25.19	LL_TIM_GetCounter.....	535
2.25.20	LL_TIM_GetDirection	535
2.25.21	LL_TIM_SetPrescaler.....	535
2.25.22	LL_TIM_GetPrescaler.....	536
2.25.23	LL_TIM_SetAutoReload.....	536
2.25.24	LL_TIM_GetAutoReload	536
2.25.25	LL_TIM_SetRepetitionCounter.....	537
2.25.26	LL_TIM_GetRepetitionCounter	537
2.25.27	LL_TIM_EnableUIFRemap	537
2.25.28	LL_TIM_DisableUIFRemap	537
2.25.29	LL_TIM_IsEnabledUIFRemap.....	538
2.25.30	LL_TIM_IsActiveFlagUIFCopy.....	538
2.25.31	LL_TIM_CC_EnablePreload.....	538
2.25.32	LL_TIM_CC_DisablePreload.....	539
2.25.33	LL_TIM_CC_IsEnabledPreload.....	539
2.25.34	LL_TIM_CC_SetUpdate	539
2.25.35	LL_TIM_CC_GetUpdate.....	539
2.25.36	LL_TIM_CC_SetDMAReqTrigger	540
2.25.37	LL_TIM_CC_GetDMAReqTrigger.....	540
2.25.38	LL_TIM_CC_SetLockLevel.....	540
2.25.39	LL_TIM_CC_GetLockLevel.....	541
2.25.40	LL_TIM_CC_EnableChannel.....	541
2.25.41	LL_TIM_CC_DisableChannel	541
2.25.42	LL_TIM_CC_IsEnabledChannel.....	542
2.25.43	LL_TIM_OC_ConfigOutput.....	542
2.25.44	LL_TIM_OC_SetMode	543
2.25.45	LL_TIM_OC_GetMode.....	543
2.25.46	LL_TIM_OC_SetPolarity	544
2.25.47	LL_TIM_OC_GetPolarity	544
2.25.48	LL_TIM_OC_SetIdleState	545
2.25.49	LL_TIM_OC_GetIdleState.....	545
2.25.50	LL_TIM_OC_EnableFast.....	546
2.25.51	LL_TIM_OC_DisableFast.....	546
2.25.52	LL_TIM_OC_IsEnabledFast	546
2.25.53	LL_TIM_OC_EnablePreload	547
2.25.54	LL_TIM_OC_DisablePreload	547
2.25.55	LL_TIM_OC_IsEnabledPreload.....	547
2.25.56	LL_TIM_OC_EnableClear	548
2.25.57	LL_TIM_OC_DisableClear.....	548
2.25.58	LL_TIM_OC_IsEnabledClear	549
2.25.59	LL_TIM_OC_SetDeadTime	549
2.25.60	LL_TIM_OC_GetDeadTime	549
2.25.61	LL_TIM_OC_SetCompareCH1	550
2.25.62	LL_TIM_OC_SetCompareCH2	550
2.25.63	LL_TIM_OC_SetCompareCH3	550
2.25.64	LL_TIM_OC_SetCompareCH4	551

2.25.65	LL_TIM_OC_SetCompareCH5	551
2.25.66	LL_TIM_OC_SetCompareCH6	551
2.25.67	LL_TIM_OC_GetCompareCH1	552
2.25.68	LL_TIM_OC_GetCompareCH2	552
2.25.69	LL_TIM_OC_GetCompareCH3	552
2.25.70	LL_TIM_OC_GetCompareCH4	552
2.25.71	LL_TIM_OC_GetCompareCH5	553
2.25.72	LL_TIM_OC_GetCompareCH6	553
2.25.73	LL_TIM_SetCH5CombinedChannels	553
2.25.74	LL_TIM_GetCH5CombinedChannels	554
2.25.75	LL_TIM_IC_Config	554
2.25.76	LL_TIM_IC_SetActiveInput	555
2.25.77	LL_TIM_IC_GetActiveInput	555
2.25.78	LL_TIM_IC_SetPrescaler	555
2.25.79	LL_TIM_IC_GetPrescaler	556
2.25.80	LL_TIM_IC_SetFilter	556
2.25.81	LL_TIM_IC_GetFilter	557
2.25.82	LL_TIM_IC_SetPolarity	557
2.25.83	LL_TIM_IC_GetPolarity	558
2.25.84	LL_TIM_IC_EnableXORCombination	558
2.25.85	LL_TIM_IC_DisableXORCombination	558
2.25.86	LL_TIM_IC_IsEnabledXORCombination	559
2.25.87	LL_TIM_IC_GetCaptureCH1	559
2.25.88	LL_TIM_IC_GetCaptureCH2	559
2.25.89	LL_TIM_IC_GetCaptureCH3	560
2.25.90	LL_TIM_IC_GetCaptureCH4	560
2.25.91	LL_TIM_EnableExternalClock	560
2.25.92	LL_TIM_DisableExternalClock	560
2.25.93	LL_TIM_IsEnabledExternalClock	561
2.25.94	LL_TIM_SetClockSource	561
2.25.95	LL_TIM_GetClockSource	561
2.25.96	LL_TIM_SetEncoderMode	562
2.25.97	LL_TIM_GetEncoderMode	562
2.25.98	LL_TIM_SetTriggerOutput	562
2.25.99	LL_TIM_GetTriggerOutput	563
2.25.100	LL_TIM_SetTriggerOutput2	563
2.25.101	LL_TIM_GetTriggerOutput2	564
2.25.102	LL_TIM_SetSlaveMode	564
2.25.103	LL_TIM_GetSlaveMode	564
2.25.104	LL_TIM_SetTriggerInput	565
2.25.105	LL_TIM_GetTriggerInput	565
2.25.106	LL_TIM_EnableMasterSlaveMode	566
2.25.107	LL_TIM_DisableMasterSlaveMode	566
2.25.108	LL_TIM_IsEnabledMasterSlaveMode	566
2.25.109	LL_TIM_ConfigETR	566
2.25.110	LL_TIM_GetETRFilter	567

2.25.111	LL_TIM_GetETRPrescale.....	568
2.25.112	LL_TIM_GetETRPolarity.....	568
2.25.113	LL_TIM_SetETRSource.....	568
2.25.114	LL_TIM_GetETRSource.....	569
2.25.115	LL_TIM_EnableBRK.....	569
2.25.116	LL_TIM_DisableBRK.....	569
2.25.117	LL_TIM_IsEnabledBRK.....	570
2.25.118	LL_TIM_ConfigBRK.....	570
2.25.119	LL_TIM_GetBRKPolarity.....	571
2.25.120	LL_TIM_GetBRKAFMode.....	571
2.25.121	LL_TIM_GetBRKFilter.....	571
2.25.122	LL_TIM_DisarmBRK.....	572
2.25.123	LL_TIM_ReArmBRK.....	572
2.25.124	LL_TIM_IsDisarmedBRK.....	572
2.25.125	LL_TIM_EnableBRK2.....	573
2.25.126	LL_TIM_DisableBRK2.....	573
2.25.127	LL_TIM_IsEnabledBRK2.....	573
2.25.128	LL_TIM_ConfigBRK2.....	574
2.25.129	LL_TIM_GetBRK2Polarity.....	574
2.25.130	LL_TIM_GetBRK2AFMode.....	575
2.25.131	LL_TIM_GetBRK2Filter.....	575
2.25.132	LL_TIM_DisarmBRK2.....	576
2.25.133	LL_TIM_ReArmBRK2.....	576
2.25.134	LL_TIM_IsDisarmedBRK2.....	576
2.25.135	LL_TIM_SetOffStates.....	576
2.25.136	LL_TIM_GetOffStates.....	577
2.25.137	LL_TIM_EnableAutomaticOutput.....	577
2.25.138	LL_TIM_DisableAutomaticOutput.....	577
2.25.139	LL_TIM_IsEnabledAutomaticOutput.....	578
2.25.140	LL_TIM_EnableAllOutputs.....	578
2.25.141	LL_TIM_DisableAllOutputs.....	578
2.25.142	LL_TIM_IsEnabledAllOutputs.....	578
2.25.143	LL_TIM_EnableBreakInputSource.....	579
2.25.144	LL_TIM_DisableBreakInputSource.....	579
2.25.145	LL_TIM_IsEnabledBreakInputSource.....	579
2.25.146	LL_TIM_SetBreakInputSourcePolarity.....	580
2.25.147	LL_TIM_GetBreakInputSourcePolarity.....	580
2.25.148	LL_TIM_ConfigDMABurst.....	581
2.25.149	LL_TIM_GetDMABurstLength.....	582
2.25.150	LL_TIM_GetDMABurstBaseAddress.....	583
2.25.151	LL_TIM_DMA_GetRegAddr.....	584
2.25.152	LL_TIM_DMA_GetRegARRAddr.....	584
2.25.153	LL_TIM_DMA_GetRegCCR1Addr.....	584
2.25.154	LL_TIM_DMA_GetRegCCR2Addr.....	584
2.25.155	LL_TIM_DMA_GetRegCCR3Addr.....	585
2.25.156	LL_TIM_DMA_GetRegCCR4Addr.....	585

2.25.157	LL_TIM_SetRemap.....	585
2.25.158	LL_TIM_SetChannelRemap.....	586
2.25.159	LL_TIM_GetRemap	586
2.25.160	LL_TIM_SetOCRefClearInputSource.....	587
2.25.161	LL_TIM_ClearFlag_UPDATE	587
2.25.162	LL_TIM_IsActiveFlag_UPDATE	588
2.25.163	LL_TIM_ClearFlag_CC1	588
2.25.164	LL_TIM_IsActiveFlag_CC1	588
2.25.165	LL_TIM_ClearFlag_CC2	588
2.25.166	LL_TIM_IsActiveFlag_CC2	589
2.25.167	LL_TIM_ClearFlag_CC3	589
2.25.168	LL_TIM_IsActiveFlag_CC3	589
2.25.169	LL_TIM_ClearFlag_CC4	589
2.25.170	LL_TIM_IsActiveFlag_CC4	590
2.25.171	LL_TIM_ClearFlag_CC5	590
2.25.172	LL_TIM_IsActiveFlag_CC5	590
2.25.173	LL_TIM_ClearFlag_CC6	591
2.25.174	LL_TIM_IsActiveFlag_CC6	591
2.25.175	LL_TIM_ClearFlag_COM	591
2.25.176	LL_TIM_IsActiveFlag_COM.....	591
2.25.177	LL_TIM_ClearFlag_TRIG	592
2.25.178	LL_TIM_IsActiveFlag_TRIG	592
2.25.179	LL_TIM_ClearFlag_BRK	592
2.25.180	LL_TIM_IsActiveFlag_BRK	592
2.25.181	LL_TIM_ClearFlag_BRK2	593
2.25.182	LL_TIM_IsActiveFlag_BRK2	593
2.25.183	LL_TIM_ClearFlag_CC1OVR.....	593
2.25.184	LL_TIM_IsActiveFlag_CC1OVR.....	594
2.25.185	LL_TIM_ClearFlag_CC2OVR.....	594
2.25.186	LL_TIM_IsActiveFlag_CC2OVR.....	594
2.25.187	LL_TIM_ClearFlag_CC3OVR.....	594
2.25.188	LL_TIM_IsActiveFlag_CC3OVR.....	595
2.25.189	LL_TIM_ClearFlag_CC4OVR.....	595
2.25.190	LL_TIM_IsActiveFlag_CC4OVR.....	595
2.25.191	LL_TIM_ClearFlag_SYSBRK.....	595
2.25.192	LL_TIM_IsActiveFlag_SYSBRK	596
2.25.193	LL_TIM_ClearFlag	596
2.25.194	LL_TIM_IsActiveFlag.....	596
2.25.195	LL_TIM_EnableIT_UPDATE	597
2.25.196	LL_TIM_DisableIT_UPDATE	597
2.25.197	LL_TIM_IsEnabledIT_UPDATE	597
2.25.198	LL_TIM_EnableIT_CC1	598
2.25.199	LL_TIM_DisableIT_CC1	598
2.25.200	LL_TIM_IsEnabledIT_CC1	598
2.25.201	LL_TIM_EnableIT_CC2	599
2.25.202	LL_TIM_DisableIT_CC2.....	599

2.25.203	LL_TIM_IsEnabledIT_CC2	599
2.25.204	LL_TIM_EnableIT_CC3	599
2.25.205	LL_TIM_DisableIT_CC3	600
2.25.206	LL_TIM_IsEnabledIT_CC3	600
2.25.207	LL_TIM_EnableIT_CC4	600
2.25.208	LL_TIM_DisableIT_CC4	600
2.25.209	LL_TIM_IsEnabledIT_CC4	601
2.25.210	LL_TIM_EnableIT_COM	601
2.25.211	LL_TIM_DisableIT_COM	601
2.25.212	LL_TIM_IsEnabledIT_COM	602
2.25.213	LL_TIM_EnableIT_TRIG	602
2.25.214	LL_TIM_DisableIT_TRIG	602
2.25.215	LL_TIM_IsEnabledIT_TRIG	602
2.25.216	LL_TIM_EnableIT_BRK	603
2.25.217	LL_TIM_DisableIT_BRK	603
2.25.218	LL_TIM_IsEnabledIT_BRK	603
2.25.219	LL_TIM_EnableIT	603
2.25.220	LL_TIM_DisableIT	604
2.25.221	LL_TIM_IsEnabledIT	604
2.25.222	LL_TIM_EnableDMAReq_UPDATE	605
2.25.223	LL_TIM_DisableDMAReq_UPDATE	605
2.25.224	LL_TIM_IsEnabledDMAReq_UPDATE	605
2.25.225	LL_TIM_DMAReq_CC1	605
2.25.226	LL_TIM_DisableDMAReq_CC1	606
2.25.227	LL_TIM_IsEnabledDMAReq_CC1	606
2.25.228	LL_TIM_EnableDMAReq_CC2	606
2.25.229	LL_TIM_DisableDMAReq_CC2	606
2.25.230	LL_TIM_IsEnabledDMAReq_CC2	607
2.25.231	LL_TIM_EnableDMAReq_CC3	607
2.25.232	LL_TIM_DisableDMAReq_CC3	607
2.25.233	LL_TIM_IsEnabledDMAReq_CC3	608
2.25.234	LL_TIM_EnableDMAReq_CC4	608
2.25.235	LL_TIM_DisableDMAReq_CC4	608
2.25.236	LL_TIM_IsEnabledDMAReq_CC4	608
2.25.237	LL_TIM_EnableDMAReq_COM	609
2.25.238	LL_TIM_DisableDMAReq_COM	609
2.25.239	LL_TIM_IsEnabledDMAReq_COM	609
2.25.240	LL_TIM_EnableDMAReq_TRIG	609
2.25.241	LL_TIM_DisableDMAReq_TRIG	610
2.25.242	LL_TIM_IsEnabledDMAReq_TRIG	610
2.25.243	LL_TIM_EnableDMAReq	610
2.25.244	LL_TIM_DisableDMAReq	611
2.25.245	LL_TIM_IsEnabledDMAReq	611
2.25.246	LL_TIM_GenerateEvent_UPDATE	611
2.25.247	LL_TIM_GenerateEvent_CC1	612
2.25.248	LL_TIM_GenerateEvent_CC2	612

2.25.249	LL_TIM_GenerateEvent_CC3	612
2.25.250	LL_TIM_GenerateEvent_CC4	612
2.25.251	LL_TIM_GenerateEvent_COM	613
2.25.252	LL_TIM_GenerateEvent_TRIG	613
2.25.253	LL_TIM_GenerateEvent_BRK	613
2.25.254	LL_TIM_GenerateEvent_BRK2	614
2.25.255	LL_TIM_GenerateEvent.....	614
2.25.256	LL_TIM_DeInit	614
2.25.257	LL_TIM_StructInit	615
2.25.258	LL_TIM_Init.....	615
2.25.259	LL_TIM_OC_StructInit.....	615
2.25.260	LL_TIM_OC_Init	615
2.25.261	LL_TIM_IC_StructInit	616
2.25.262	LL_TIM_IC_Init.....	616
2.25.263	LL_TIM_ENCODER_StructInit	616
2.25.264	LL_TIM_ENCODER_Init.....	617
2.25.265	LL_TIM_HALLSENSOR_StructInit	617
2.25.266	LL_TIM_HALLSENSOR_Init.....	617
2.25.267	LL_TIM_BDTR_StructInit.....	618
2.25.268	LL_TIM_BDTR_Init	618
2.26	USART 模块.....	618
2.26.1	LL_USART_Enable.....	619
2.26.2	LL_USART_Disable	620
2.26.3	LL_USART_IsEnabled.....	620
2.26.4	LL_USART_EnableFIFO	620
2.26.5	LL_USART_DisableFIFO	620
2.26.6	LL_USART_IsEnabledFIFO	621
2.26.7	LL_USART_SetTXFIFOThreshold	621
2.26.8	LL_USART_GetTXFIFOThreshold.....	621
2.26.9	LL_USART_SetRXFIFOThreshold	622
2.26.10	LL_USART_GetRXFIFOThreshold.....	622
2.26.11	LL_USART_ConfigFIFOsThreshold	623
2.26.12	LL_USART_EnableInStopMode.....	623
2.26.13	LL_USART_DisableInStopMode.....	623
2.26.14	LL_USART_IsEnabledInStopMode.....	624
2.26.15	LL_USART_EnableDirectionRx	624
2.26.16	LL_USART_DisableDirectionRx	624
2.26.17	LL_USART_IsEnabledDirectionRx	625
2.26.18	LL_USART_EnableDirectionTx	625
2.26.19	LL_USART_DisableDirectionTx	625
2.26.20	LL_USART_IsEnabledDirectionTx	625
2.26.21	LL_USART_SetTransferDirection	626
2.26.22	LL_USART_GetTransferDirection.....	626
2.26.23	LL_USART_SetParity	626
2.26.24	LL_USART_GetParity	627

2.26.25	LL_USART_SetWakeUpMethod	627
2.26.26	LL_USART_GetWakeUpMethod.....	627
2.26.27	LL_USART_SetDataWidth	628
2.26.28	LL_USART_GetDataWidth.....	628
2.26.29	LL_USART_EnableMuteMode.....	628
2.26.30	LL_USART_DisableMuteMode.....	629
2.26.31	LL_USART_IsEnabledMuteMode	629
2.26.32	LL_USART_SetOverSampling	629
2.26.33	LL_USART_GetOverSampling.....	629
2.26.34	LL_USART_SetLastClkPulseOutput	630
2.26.35	LL_USART_GetLastClkPulseOutput.....	630
2.26.36	LL_USART_SetClockPhase.....	630
2.26.37	LL_USART_GetClockPhase	631
2.26.38	LL_USART_SetClockPolarity	631
2.26.39	LL_USART_GetClockPolarity	631
2.26.40	LL_USART_ConfigClock	632
2.26.41	LL_USART_SetPrescaler.....	632
2.26.42	LL_USART_GetPrescaler	633
2.26.43	LL_USART_EnableSCLKOutput	633
2.26.44	LL_USART_DisableSCLKOutput	633
2.26.45	LL_USART_IsEnabledSCLKOutput.....	633
2.26.46	LL_USART_SetStopBitsLength.....	634
2.26.47	LL_USART_GetStopBitsLength.....	634
2.26.48	LL_USART_ConfigCharacter	634
2.26.49	LL_USART_SetTXRXSwap.....	635
2.26.50	LL_USART_GetTXRXSwap	635
2.26.51	LL_USART_SetRXPinLevel.....	635
2.26.52	LL_USART_GetRXPinLevel	636
2.26.53	LL_USART_SetTXPinLevel	636
2.26.54	LL_USART_GetTXPinLevel	636
2.26.55	LL_USART_SetBinaryDataLogic	637
2.26.56	LL_USART_GetBinaryDataLogic	637
2.26.57	LL_USART_SetTransferBitOrder	637
2.26.58	LL_USART_GetTransferBitOrder	638
2.26.59	LL_USART_EnableAutoBaudRate	638
2.26.60	LL_USART_DisableAutoBaudRate.....	638
2.26.61	LL_USART_IsEnabledAutoBaud	639
2.26.62	LL_USART_SetAutoBaudRateMode.....	639
2.26.63	LL_USART_GetAutoBaudRateMode.....	639
2.26.64	LL_USART_EnableRxTimeout.....	640
2.26.65	LL_USART_DisableRxTimeout.....	640
2.26.66	LL_USART_IsEnabledRxTimeout.....	640
2.26.67	LL_USART_ConfigNodeAddress	640
2.26.68	LL_USART_GetNodeAddress	641
2.26.69	LL_USART_GetNodeAddressLen	641
2.26.70	LL_USART_EnableRTSHWFlowCtrl.....	641

2.26.71	LL_USART_DisableRTSHWFlowCtrl	642
2.26.72	LL_USART_EnableCTSHWFlowCtrl	642
2.26.73	LL_USART_DisableCTSHWFlowCtrl	642
2.26.74	LL_USART_SetHWFlowCtrl	643
2.26.75	LL_USART_GetHWFlowCtrl	643
2.26.76	LL_USART_EnableOneBitSamp	643
2.26.77	LL_USART_DisableOneBitSamp	644
2.26.78	LL_USART_IsEnabledOneBitSamp	644
2.26.79	LL_USART_EnableOverrunDetect	644
2.26.80	LL_USART_DisableOverrunDetect	644
2.26.81	LL_USART_IsEnabledOverrunDetect	645
2.26.82	LL_USART_SetWKUPType	645
2.26.83	LL_USART_GetWKUPType	645
2.26.84	LL_USART_SetBaudRate	646
2.26.85	LL_USART_GetBaudRate	646
2.26.86	LL_USART_SetRxTimeout	647
2.26.87	LL_USART_GetRxTimeout	647
2.26.88	LL_USART_SetBlockLength	647
2.26.89	LL_USART_GetBlockLength	648
2.26.90	LL_USART_EnableIrda	648
2.26.91	LL_USART_DisableIrda	648
2.26.92	LL_USART_IsEnabledIrda	648
2.26.93	LL_USART_SetIrdaPowerMode	649
2.26.94	LL_USART_GetIrdaPowerMode	649
2.26.95	LL_USART_SetIrdaPrescaler	649
2.26.96	LL_USART_GetIrdaPrescaler	650
2.26.97	LL_USART_EnableHalfDuplex	650
2.26.98	LL_USART_DisableHalfDuplex	650
2.26.99	LL_USART_IsEnabledHalfDuplex	650
2.26.100	LL_USART_EnableSPISlave	651
2.26.101	LL_USART_DisableSPISlave	651
2.26.102	LL_USART_IsEnabledSPISlave	651
2.26.103	LL_USART_EnableSPISlaveSelect	652
2.26.104	LL_USART_DisableSPISlaveSelect	652
2.26.105	LL_USART_IsEnabledSPISlaveSelect	652
2.26.106	LL_USART_SetDEDeassertionTime	652
2.26.107	LL_USART_GetDEDeassertionTime	653
2.26.108	LL_USART_SetDEAssertionTime	653
2.26.109	LL_USART_GetDEAssertionTime	653
2.26.110	LL_USART_EnableDEMode	654
2.26.111	LL_USART_DisableDEMode	654
2.26.112	LL_USART_IsEnabledDEMode	654
2.26.113	LL_USART_SetDESignalPolarity	654
2.26.114	LL_USART_GetDESignalPolarity	655
2.26.115	LL_USART_ConfigAsyncMode	655
2.26.116	LL_USART_ConfigSyncMode	655

2.26.117	LL_USART_ConfigHalfDuplexMode.....	656
2.26.118	LL_USART_ConfigIrdaMode.....	656
2.26.119	LL_USART_ConfigMultiProcessMode	656
2.26.120	LL_USART_IsActiveFlag_PE.....	656
2.26.121	LL_USART_IsActiveFlag_FE.....	657
2.26.122	LL_USART_IsActiveFlag_NE.....	657
2.26.123	LL_USART_IsActiveFlag_ORE	657
2.26.124	LL_USART_IsActiveFlag_IDLE.....	658
2.26.125	LL_USART_IsActiveFlag_RXNE_RXFNE	658
2.26.126	LL_USART_IsActiveFlag_TC	658
2.26.127	LL_USART_IsActiveFlag_TXE_TXFNF.....	658
2.26.128	LL_USART_IsActiveFlag_nCTS.....	659
2.26.129	LL_USART_IsActiveFlag_CTS.....	659
2.26.130	LL_USART_IsActiveFlag_RTO	659
2.26.131	LL_USART_IsActiveFlag_UDR.....	659
2.26.132	LL_USART_IsActiveFlag_ABRE.....	660
2.26.133	LL_USART_IsActiveFlag_ABR.....	660
2.26.134	LL_USART_IsActiveFlag_BUSY.....	660
2.26.135	LL_USART_IsActiveFlag_CM.....	661
2.26.136	LL_USART_IsActiveFlag_SBK	661
2.26.137	LL_USART_IsActiveFlag_RWU.....	661
2.26.138	LL_USART_IsActiveFlag_WKUP	661
2.26.139	LL_USART_IsActiveFlag_TEACK.....	662
2.26.140	LL_USART_IsActiveFlag_REACK.....	662
2.26.141	LL_USART_IsActiveFlag_TXFE	662
2.26.142	LL_USART_IsActiveFlag_RXFE	663
2.26.143	LL_USART_IsActiveFlag_TXFT	663
2.26.144	LL_USART_IsActiveFlag_RXFT	663
2.26.145	LL_USART_IsActiveFlag.....	663
2.26.146	LL_USART_ClearFlag_PE	664
2.26.147	LL_USART_ClearFlag_FE	664
2.26.148	LL_USART_ClearFlag_NE.....	665
2.26.149	LL_USART_ClearFlag_ORE	665
2.26.150	LL_USART_ClearFlag_IDLE.....	665
2.26.151	LL_USART_ClearFlag_TC.....	665
2.26.152	LL_USART_ClearFlag_nCTS.....	666
2.26.153	LL_USART_ClearFlag_CM.....	666
2.26.154	LL_USART_ClearFlag_WKUP	666
2.26.155	LL_USART_ClearFlag_RTO	666
2.26.156	LL_USART_ClearFlag_UDR.....	667
2.26.157	LL_USART_ClearFlag.....	667
2.26.158	LL_USART_EnableIT_IDLE.....	667
2.26.159	LL_USART_EnableIT_RXNE_RXFNE.....	668
2.26.160	LL_USART_EnableIT_TC.....	668
2.26.161	LL_USART_EnableIT_TXE_TXFNF.....	668
2.26.162	LL_USART_EnableIT_PE	669

2.26.163	LL_USART_EnableIT_CM.....	669
2.26.164	LL_USART_EnableIT_RTO	669
2.26.165	LL_USART_EnableIT_TXFE.....	669
2.26.166	LL_USART_EnableIT_RXFF.....	670
2.26.167	LL_USART_EnableIT_ERROR.....	670
2.26.168	LL_USART_EnableIT_CTS.....	670
2.26.169	LL_USART_EnableIT_WKUP	670
2.26.170	LL_USART_EnableIT_TXFT.....	671
2.26.171	LL_USART_EnableIT_RXFT.....	671
2.26.172	LL_USART_DisableIT_IDLE.....	671
2.26.173	LL_USART_DisableIT_RXNE_RXFNE.....	672
2.26.174	LL_USART_DisableIT_TC.....	672
2.26.175	LL_USART_DisableIT_TXE_TXFNF	672
2.26.176	LL_USART_DisableIT_PE	672
2.26.177	LL_USART_DisableIT_CM.....	673
2.26.178	LL_USART_DisableIT_RTO	673
2.26.179	LL_USART_DisableIT_TXFE.....	673
2.26.180	LL_USART_DisableIT_RXFF	673
2.26.181	LL_USART_DisableIT_ERROR	674
2.26.182	LL_USART_DisableIT_CTS	674
2.26.183	LL_USART_DisableIT_WKUP.....	674
2.26.184	LL_USART_DisableIT_TXFT.....	675
2.26.185	LL_USART_DisableIT_RXFT.....	675
2.26.186	LL_USART_IsEnabledIT_IDLE.....	675
2.26.187	LL_USART_IsEnabledIT_RXNE_RXFNE.....	675
2.26.188	LL_USART_IsEnabledIT_TC.....	676
2.26.189	LL_USART_IsEnabledIT_TXE_TXFNF.....	676
2.26.190	LL_USART_IsEnabledIT_PE	676
2.26.191	LL_USART_IsEnabledIT_CM.....	676
2.26.192	LL_USART_IsEnabledIT_RTO	677
2.26.193	LL_USART_IsEnabledIT_TXFE.....	677
2.26.194	LL_USART_IsEnabledIT_RXFF	677
2.26.195	LL_USART_IsEnabledIT_ERROR.....	678
2.26.196	LL_USART_IsEnabledIT_CTS.....	678
2.26.197	LL_USART_IsEnabledIT_WKUP	678
2.26.198	LL_USART_IsEnabledIT_TXFT	678
2.26.199	LL_USART_IsEnabledIT_RXFT.....	679
2.26.200	LL_USART_EnableDMAReq_RX	679
2.26.201	LL_USART_DisableDMAReq_RX	679
2.26.202	LL_USART_IsEnabledDMAReq_RX.....	679
2.26.203	LL_USART_EnableDMAReq_TX.....	680
2.26.204	LL_USART_DisableDMAReq_TX.....	680
2.26.205	LL_USART_IsEnabledDMAReq_TX.....	680
2.26.206	LL_USART_EnableDMADeactOnRxErr	681
2.26.207	LL_USART_DisableDMADeactOnRxErr	681
2.26.208	LL_USART_IsEnabledDMADeactOnRxErr.....	681

2.26.209	LL_USART_DMA_GetRegAddr	681
2.26.210	LL_USART_ReceiveData8	682
2.26.211	LL_USART_ReceiveData9	682
2.26.212	LL_USART_TransmitData8	682
2.26.213	LL_USART_TransmitData9	683
2.26.214	LL_USART_RequestAutoBaudRate	683
2.26.215	LL_USART_RequestBreakSending	683
2.26.216	LL_USART_RequestEnterMuteMode	684
2.26.217	LL_USART_RequestRxDataFlush	684
2.26.218	LL_USART_RequestTxDataFlush	684
2.26.219	LL_USART_DeInit	684
2.26.220	LL_USART_Init	685
2.26.221	LL_USART_StructInit	685
2.26.222	LL_USART_ClockInit	685
2.26.223	LL_USART_ClockStructInit	686
2.27	UTILS 模块	686
2.27.1	LL_GetUID_Word0	686
2.27.2	LL_GetUID_Word1	687
2.27.3	LL_GetUID_Word2	687
2.27.4	LL_GetFlashSize	687
2.27.5	LL_GetSRAMSize	687
2.27.6	LL_GetPackageType	688
2.27.7	LL_GetDeviceType	688
2.27.8	LL_InitTick	688
2.27.9	LL_Init1msTick	689
2.27.10	LL_mDelay	689
2.27.11	LL_SetSystemCoreClock	689
2.27.12	LL_PLL_ConfigSystemClock_HSI	689
2.27.13	LL_PLL_ConfigSystemClock_HSE	690
2.27.14	LL_SetFlashLatency	690
2.28	WWDG 模块	690
2.28.1	LL_WWDG_Enable	691
2.28.2	LL_WWDG_IsEnabled	691
2.28.3	LL_WWDG_SetCounter	691
2.28.4	LL_WWDG_GetCounter	691
2.28.5	LL_WWDG_SetPrescaler	692
2.28.6	LL_WWDG_GetPrescaler	692
2.28.7	LL_WWDG_SetWindow	692
2.28.8	LL_WWDG_GetWindow	693
2.28.9	LL_WWDG_IsActiveFlag_EWKUP	693
2.28.10	LL_WWDG_ClearFlag_EWKUP	693
2.28.11	LL_WWDG_EnableIT_EWKUP	694
2.28.12	LL_WWDG_IsEnabledIT_EWKUP	694
2.29	BEMF 模块	694
2.29.1	LL_BEMF_SetUSEL	695

2.29.2	LL_BEMF_GetUSEL.....	695
2.29.3	LL_BEMF_SetVSEL.....	695
2.29.4	LL_BEMF_GetVSEL.....	696
2.29.5	LL_BEMF_SetWSEL.....	696
2.29.6	LL_BEMF_GetWSEL.....	696
2.29.7	LL_BEMF_SetMIDSEL.....	696
2.29.8	LL_BEMF_GetMIDSEL.....	697
2.29.9	LL_BEMF_DeInit.....	697
2.29.10	LL_BEMF_Init.....	697
2.29.11	LL_BEMF_StructInit.....	698
2.30	CORDIC 模块.....	698
2.30.1	LL_CORDIC_Config.....	698
2.30.2	LL_CORDIC_SetFunction.....	699
2.30.3	LL_CORDIC_GetFunction.....	699
2.30.4	LL_CORDIC_SetPrecision.....	700
2.30.5	LL_CORDIC_GetPrecision.....	700
2.30.6	LL_CORDIC_SetNbWrite.....	701
2.30.7	LL_CORDIC_GetNbWrite.....	701
2.30.8	LL_CORDIC_SetNbRead.....	702
2.30.9	LL_CORDIC_GetNbRead.....	702
2.30.10	LL_CORDIC_SetInSize.....	702
2.30.11	LL_CORDIC_GetInSize.....	702
2.30.12	LL_CORDIC_SetOutSize.....	703
2.30.13	LL_CORDIC_GetOutSize.....	703
2.30.14	LL_CORDIC_EnableIT.....	703
2.30.15	LL_CORDIC_DisableIT.....	704
2.30.16	LL_CORDIC_IsEnabledIT.....	704
2.30.17	LL_CORDIC_EnableDMAReq_RD.....	704
2.30.18	LL_CORDIC_DisableDMAReq_RD.....	704
2.30.19	LL_CORDIC_IsEnabledDMAReq_RD.....	705
2.30.20	LL_CORDIC_EnableDMAReq_WR.....	705
2.30.21	LL_CORDIC_DisableDMAReq_WR.....	705
2.30.22	LL_CORDIC_IsEnabledDMAReq_WR.....	706
2.30.23	LL_CORDIC_DMA_GetRegAddr.....	706
2.30.24	LL_CORDIC_IsActiveFlag_RRDY.....	706
2.30.25	LL_CORDIC_WriteData.....	706
2.30.26	LL_CORDIC_ReadData.....	707
2.30.27	LL_CORDIC_DeInit.....	707
3	API (基础 HAL 层).....	708
3.1	IWDG 模块.....	708
3.1.1	HAL_IWDG_Refresh.....	708
3.1.2	HAL_IWDG_Init.....	709
3.2	CORTEX 模块.....	709
3.2.1	HAL_NVIC_SetPriority.....	710
3.2.2	HAL_NVIC_EnableIRQ.....	711

3.2.3	HAL_NVIC_DisableIRQ	711
3.2.4	HAL_NVIC_SystemReset.....	711
3.2.5	HAL_SYSTICK_Config	711
3.2.6	HAL_NVIC_GetPriority	712
3.2.7	HAL_NVIC_GetPendingIRQ.....	712
3.2.8	HAL_NVIC_SetPendingIRQ	712
3.2.9	HAL_NVIC_ClearPendingIRQ.....	713
3.2.10	HAL_SYSTICK_CLKSourceConfig	713
3.2.11	HAL_SYSTICK_IRQHandler.....	713
3.2.12	HAL_SYSTICK_Callback	713
3.3	EXTI 模块	714
3.3.1	HAL_EXTI_SetConfigLine.....	714
3.3.2	HAL_EXTI_GetConfigLine	714
3.3.3	HAL_EXTI_ClearConfigLine	715
3.3.4	HAL_EXTI_RegisterCallback.....	715
3.3.5	HAL_EXTI_GetHandle.....	715
3.3.6	HAL_EXTI_IRQHandler	716
3.3.7	HAL_EXTI_GetPending	716
3.3.8	HAL_EXTI_ClearPending	716
3.4	FLASH 模块.....	717
3.4.1	HAL_FLASH_Program.....	718
3.4.2	HAL_FLASH_Program_IT	718
3.4.3	HAL_FLASH_ProgramHalfWord	718
3.4.4	HAL_FLASH_ProgramHalfWord_IT	719
3.4.5	HAL_FLASH_IRQHandler.....	719
3.4.6	HAL_FLASH_EndOfOperationCallback.....	719
3.4.7	HAL_FLASH_OperationErrorCallback	720
3.4.8	HAL_FLASH_Unlock.....	720
3.4.9	HAL_FLASH_Lock	720
3.4.10	HAL_FLASH_OB_Unlock	720
3.4.11	HAL_FLASH_OB_Lock.....	721
3.4.12	HAL_FLASH_OB_Launch.....	721
3.4.13	HAL_FLASH_GetError	721
3.4.14	HAL_FLASHEx_Erase	722
3.4.15	HAL_FLASHEx_Erase_IT.....	722
3.4.16	HAL_FLASHEx_EnableSecMemProtection.....	722
3.4.17	HAL_FLASHEx_OBProgram.....	722
3.4.18	HAL_FLASHEx_OBGetConfig.....	723
3.5	GPIO 模块	723
3.5.1	HAL_GPIO_Init	724
3.5.2	HAL_GPIO_DeInit.....	724
3.5.3	HAL_GPIO_ReadPin	725
3.5.4	HAL_GPIO_WritePin.....	725
3.5.5	HAL_GPIO_TogglePin.....	726
3.5.6	HAL_GPIO_LockPin	726

3.5.7	HAL_GPIO_EXTI_IRQHandler	727
3.5.8	HAL_GPIO_EXTI_Rising_Callback	727
3.5.9	HAL_GPIO_EXTI_Falling_Callback	727
3.6	PWR 模块	728
3.6.1	HAL_PWR_DeInit	728
3.6.2	HAL_PWR_EnterSLEEPMode	728
3.6.3	HAL_PWR_EnterSTOPMode	729
3.6.4	HAL_PWR_EnableSleepOnExit	729
3.6.5	HAL_PWR_DisableSleepOnExit	729
3.6.6	HAL_PWR_EnableSEVOnPend	730
3.6.7	HAL_PWR_DisableSEVOnPend	730
3.6.8	HAL_PWR_VREF_VoltageScalingConfig	730
3.6.9	HAL_PWR_EnableVREF	731
3.6.10	HAL_PWR_DisableVREF	731
3.6.11	HAL_PWREx_PVD_Config	731
3.6.12	HAL_PWREx_PVD_Enable	731
3.6.13	HAL_PWREx_PVD_Disable	732
3.6.14	HAL_PWREx_PVD_IRQHandler	732
3.6.15	HAL_PWREx_PVD_Callback	732
3.6.16	HAL_PWREx_EnableLowPowerRunMode	732
3.6.17	HAL_PWREx_DisableLowPowerRunMode	733
3.7	RTC 模块	733
3.7.1	HAL_RTC_Init	735
3.7.2	HAL_RTC_DeInit	735
3.7.3	HAL_RTC_MspInit	736
3.7.4	HAL_RTC_MspDeInit	736
3.7.5	HAL_RTC_SetTime	736
3.7.6	HAL_RTC_GetTime	736
3.7.7	HAL_RTC_SetDate	737
3.7.8	HAL_RTC_GetDate	737
3.7.9	HAL_RTC_SetAlarm	738
3.7.10	HAL_RTC_SetAlarm_IT	738
3.7.11	HAL_RTC_DeactivateAlarm	738
3.7.12	HAL_RTC_GetAlarm	739
3.7.13	HAL_RTC_SetPeriod_IT	739
3.7.14	HAL_RTC_IRQHandler	739
3.7.15	HAL_RTC_PollForAlarmEvent	739
3.7.16	HAL_RTC_AlarmEventCallback	740
3.7.17	HAL_RTC_PeriodEventCallback	740
3.7.18	HAL_RTC_SetSmoothCalib	740
3.7.19	HAL_RTC_EnterLowpowerMode	741
3.7.20	HAL_RTC_ExitLowpowerMode	741
3.7.21	HAL_RTC_GetState	741
3.8	ADC 模块	742
3.8.1	HAL_ADC_Init	743

3.8.2	HAL_ADC_DeInit.....	744
3.8.3	HAL_ADC_MspInit.....	744
3.8.4	HAL_ADC_MspDeInit.....	744
3.8.5	HAL_ADC_Start.....	744
3.8.6	HAL_ADC_Stop.....	745
3.8.7	HAL_ADC_PollForConversion.....	745
3.8.8	HAL_ADC_PollForEvent.....	745
3.8.9	HAL_ADC_Start_IT.....	746
3.8.10	HAL_ADC_Stop_IT.....	746
3.8.11	HAL_ADC_Start_DMA.....	746
3.8.12	HAL_ADC_Stop_DMA.....	747
3.8.13	HAL_ADC_GetValue.....	747
3.8.14	HAL_ADC_IRQHandler.....	747
3.8.15	HAL_ADC_ConvCpltCallback.....	747
3.8.16	HAL_ADC_ConvHalfCpltCallback.....	748
3.8.17	HAL_ADC_LevelOutOfWindowCallback.....	748
3.8.18	HAL_ADC_ErrorCallback.....	748
3.8.19	HAL_ADC_ConfigChannel.....	748
3.8.20	HAL_ADC_AnalogWDGConfig.....	749
3.8.21	HAL_ADC_GetState.....	749
3.8.22	HAL_ADC_GetError.....	749
3.8.23	HAL_ADCEx_Calibration_Start.....	750
3.8.24	HAL_ADCEx_Calibration_GetValue.....	750
3.8.25	HAL_ADCEx_Calibration_SetValue.....	750
3.8.26	HAL_ADCEx_ChannelConfigReadyCallback.....	751
3.8.27	HAL_ADCEx_ChannelCalibrationReadyCallback.....	751
3.9	ATK 模块.....	751
3.9.1	HAL_ATK_Init.....	754
3.9.2	HAL_ATK_DeInit.....	754
3.9.3	HAL_ATK_MspInit.....	754
3.9.4	HAL_ATK_MspDeInit.....	754
3.9.5	HAL_ATK_Start.....	755
3.9.6	HAL_ATK_Stop.....	755
3.9.7	HAL_ATK_Start_IT.....	755
3.9.8	HAL_ATK_Stop_IT.....	756
3.9.9	HAL_ATK_Start_DMA.....	756
3.9.10	HAL_ATK_Stop_DMA.....	756
3.9.11	HAL_ATK_GetValue.....	756
3.9.12	HAL_ATK_ConfigChannel.....	757
3.9.13	HAL_ATK_ConfigFirstChannel.....	757
3.9.14	HAL_ATK_IRQHandler.....	757
3.9.15	HAL_ATK_ConvCpltCallback.....	758
3.9.16	HAL_ATK_ConvHalfCpltCallback.....	758
3.9.17	HAL_ATK_ErrorCallback.....	758
3.9.18	HAL_ATK_GetState.....	759

3.10	COMP 模块	759
3.10.1	HAL_COMP_Init	760
3.10.2	HAL_COMP_DeInit	760
3.10.3	HAL_COMP_MspInit	761
3.10.4	HAL_COMP_MspDeInit	761
3.10.5	HAL_COMP_Start	761
3.10.6	HAL_COMP_Stop	762
3.10.7	HAL_COMP_IRQHandler	762
3.10.8	HAL_COMP_Lock	762
3.10.9	HAL_COMP_GetOutputLevel	762
3.10.10	HAL_COMP_TriggerCallback	763
3.10.11	HAL_COMP_GetState	763
3.10.12	HAL_COMP_GetError	763
3.11	CRC 模块	764
3.11.1	HAL_CRC_Init	764
3.11.2	HAL_CRC_DeInit	764
3.11.3	HAL_CRC_MspInit	765
3.11.4	HAL_CRC_MspDeInit	765
3.11.5	HAL_CRC_Accumulate	765
3.11.6	HAL_CRC_Calculate	766
3.11.7	HAL_CRC_CompareCalculate	766
3.11.8	HAL_CRC_GetState	766
3.12	DAC 模块	767
3.12.1	HAL_DAC_Init	771
3.12.2	HAL_DAC_DeInit	771
3.12.3	HAL_DAC_MspInit	771
3.12.4	HAL_DAC_MspDeInit	772
3.12.5	HAL_DAC_Start	772
3.12.6	HAL_DAC_Stop	772
3.12.7	HAL_DAC_Start_DMA	773
3.12.8	HAL_DAC_Stop_DMA	773
3.12.9	HAL_DAC_IRQHandler	774
3.12.10	HAL_DAC_SetValue	774
3.12.11	HAL_DAC_ConvCpltCallbackCh1	774
3.12.12	HAL_DAC_ConvHalfCpltCallbackCh1	775
3.12.13	HAL_DAC_ErrorCallbackCh1	775
3.12.14	HAL_DAC_DMAUnderrunCallbackCh1	775
3.12.15	HAL_DAC_GetValue	775
3.12.16	HAL_DAC_ConfigChannel	776
3.12.17	HAL_DAC_GetState	776
3.12.18	HAL_DAC_GetError	776
3.12.19	HAL_DACEx_TriangleWaveGenerate	777
3.12.20	HAL_DACEx_NoiseWaveGenerate	777
3.12.21	HAL_DACEx_DualSetValue	778
3.12.22	HAL_DACEx_DualGetValue	778

3.12.23	HAL_DACEx_ConvCpltCallbackCh2	779
3.12.24	HAL_DACEx_ConvHalfCpltCallbackCh2	779
3.12.25	HAL_DACEx_ErrorCallbackCh2	779
3.12.26	HAL_DACEx_DMAUnderrunCallbackCh2	779
3.12.27	HAL_DACEx_ConvCpltCallbackCh3	780
3.12.28	HAL_DACEx_ConvHalfCpltCallbackCh3	780
3.12.29	HAL_DACEx_ErrorCallbackCh3	780
3.12.30	HAL_DACEx_DMAUnderrunCallbackCh3	781
3.12.31	HAL_DACEx_ConvCpltCallbackCh4	781
3.12.32	HAL_DACEx_ConvHalfCpltCallbackCh4	781
3.12.33	HAL_DACEx_ErrorCallbackCh4	781
3.12.34	HAL_DACEx_DMAUnderrunCallbackCh4	782
3.12.35	HAL_DACEx_DMAUnderrunCallbackCh4	782
3.12.36	HAL_DACEx_SelfCalibrate	782
3.12.37	HAL_DACEx_SetUserTrimming	783
3.12.38	HAL_DACEx_GetTrimOffset	783
3.13	DMA 模块	783
3.13.1	HAL_DMA_Init	787
3.13.2	HAL_DMA_DeInit	787
3.13.3	HAL_DMA_Start	787
3.13.4	HAL_DMA_Start_IT	788
3.13.5	HAL_DMA_Abort	788
3.13.6	HAL_DMA_Abort_IT	788
3.13.7	HAL_DMA_PollForTransfer	788
3.13.8	HAL_DMA_IRQHandler	789
3.13.9	HAL_DMA_RegisterCallback	789
3.13.10	HAL_DMA_UnRegisterCallback	789
3.13.11	HAL_DMA_GetState	790
3.13.12	HAL_DMA_GetError	790
3.13.13	HAL_DMAEx_ConfigMuxRequestGenerator	790
3.13.14	HAL_DMAEx_EnableMuxRequestGenerator	791
3.13.15	HAL_DMAEx_DisableMuxRequestGenerator	791
3.13.16	HAL_DMAEx_ConfigMux	791
3.13.17	HAL_DMAEx_MUX_IRQHandler	792
3.14	HDIV 模块	792
3.14.1	HAL_HDIV_Init	793
3.14.2	HAL_HDIV_DeInit	793
3.14.3	HAL_HDIV_MspInit	793
3.14.4	HAL_HDIV_MspDeInit	793
3.14.5	HAL_HDIV_Calculate	794
3.14.6	HAL_HDIV_GetState	794
3.15	I2C 模块	795
3.15.1	HAL_I2C_Init	798
3.15.2	HAL_I2C_DeInit	799
3.15.3	HAL_I2C_MspInit	799

3.15.4	HAL_I2C_MspDeInit.....	799
3.15.5	HAL_I2C_Master_Transmit.....	799
3.15.6	HAL_I2C_Master_Receive.....	800
3.15.7	HAL_I2C_Slave_Transmit.....	800
3.15.8	HAL_I2C_Slave_Receive.....	801
3.15.9	HAL_I2C_Mem_Write.....	801
3.15.10	HAL_I2C_Mem_Read.....	801
3.15.11	HAL_I2C_IsDeviceReady.....	802
3.15.12	HAL_I2C_Master_Transmit_IT.....	802
3.15.13	HAL_I2C_Master_Receive_IT.....	803
3.15.14	HAL_I2C_Slave_Transmit_IT.....	803
3.15.15	HAL_I2C_Slave_Receive_IT.....	803
3.15.16	HAL_I2C_Mem_Write_IT.....	804
3.15.17	HAL_I2C_Mem_Read_IT.....	804
3.15.18	HAL_I2C_Master_Seq_Transmit_IT.....	804
3.15.19	HAL_I2C_Master_Seq_Receive_IT.....	805
3.15.20	HAL_I2C_Slave_Seq_Transmit_IT.....	805
3.15.21	HAL_I2C_Slave_Seq_Receive_IT.....	806
3.15.22	HAL_I2C_EnableListen_IT.....	806
3.15.23	HAL_I2C_DisableListen_IT.....	807
3.15.24	HAL_I2C_Master_Abort_IT.....	807
3.15.25	HAL_I2C_Master_Transmit_DMA.....	807
3.15.26	HAL_I2C_Master_Receive_DMA.....	808
3.15.27	HAL_I2C_Slave_Transmit_DMA.....	808
3.15.28	HAL_I2C_Slave_Receive_DMA.....	808
3.15.29	HAL_I2C_Mem_Write_DMA.....	809
3.15.30	HAL_I2C_Mem_Read_DMA.....	809
3.15.31	HAL_I2C_Master_Seq_Transmit_DMA.....	809
3.15.32	HAL_I2C_Master_Seq_Receive_DMA.....	810
3.15.33	HAL_I2C_Slave_Seq_Transmit_DMA.....	810
3.15.34	HAL_I2C_Slave_Seq_Receive_DMA.....	811
3.15.35	HAL_I2C_EV_IRQHandler.....	811
3.15.36	HAL_I2C_ER_IRQHandler.....	812
3.15.37	HAL_I2C_MasterTxCpltCallback.....	812
3.15.38	HAL_I2C_MasterRxCpltCallback.....	812
3.15.39	HAL_I2C_SlaveTxCpltCallback.....	812
3.15.40	HAL_I2C_SlaveRxCpltCallback.....	813
3.15.41	HAL_I2C_AddrCallback.....	813
3.15.42	HAL_I2C_ListenCpltCallback.....	813
3.15.43	HAL_I2C_MemTxCpltCallback.....	814
3.15.44	HAL_I2C_MemRxCpltCallback.....	814
3.15.45	HAL_I2C_ErrorCallback.....	814
3.15.46	HAL_I2C_AbortCpltCallback.....	814
3.15.47	HAL_I2C_GetState.....	815
3.15.48	HAL_I2C_GetMode.....	815
3.15.49	HAL_I2C_GetError.....	815

3.15.50	HAL_I2CEx_ConfigDigitalFilter	815
3.15.51	HAL_I2CEx_EnableWakeUp	816
3.15.52	HAL_I2CEx_DisableWakeUp	816
3.16	IRDA 模块	816
3.16.1	HAL_IRDA_Init	819
3.16.2	HAL_IRDA_DeInit	819
3.16.3	HAL_IRDA_MspInit	820
3.16.4	HAL_IRDA_DeInit	820
3.16.5	HAL_IRDA_Transmit	820
3.16.6	HAL_IRDA_Receive	821
3.16.7	HAL_IRDA_Transmit_IT	821
3.16.8	HAL_IRDA_Receive_IT	821
3.16.9	HAL_IRDA_Transmit_DMA	822
3.16.10	HAL_IRDA_Receive_DMA	822
3.16.11	HAL_IRDA_DMABase	822
3.16.12	HAL_IRDA_DMAResume	823
3.16.13	HAL_IRDA_DMABase	823
3.16.14	HAL_IRDA_Abort	823
3.16.15	HAL_IRDA_AbortTransmit	823
3.16.16	HAL_IRDA_AbortReceive	824
3.16.17	HAL_IRDA_Abort_IT	824
3.16.18	HAL_IRDA_AbortTransmit_IT	824
3.16.19	HAL_IRDA_AbortReceive_IT	825
3.16.20	HAL_IRDA_IRQHandler	825
3.16.21	HAL_IRDA_TxCpltCallback	825
3.16.22	HAL_IRDA_RxCpltCallback	825
3.16.23	HAL_IRDA_TxHalfCpltCallback	826
3.16.24	HAL_IRDA_RxHalfCpltCallback	826
3.16.25	HAL_IRDA_ErrorCallback	826
3.16.26	HAL_IRDA_AbortCpltCallback	827
3.16.27	HAL_IRDA_AbortTransmitCpltCallback	827
3.16.28	HAL_IRDA_AbortReceiveCpltCallback	827
3.16.29	HAL_IRDA_GetState	827
3.16.30	HAL_IRDA_GetError	828
3.17	LCDLED 模块	828
3.17.1	HAL_LCDLED_Init	829
3.17.2	HAL_LCDLED_DeInit	829
3.17.3	HAL_LCDLED_MspInit	830
3.17.4	HAL_LCDLED_MspDeInit	830
3.17.5	HAL_LCDLED_Display	830
3.17.6	HAL_LCDLED_Hold	830
3.17.7	HAL_LCDLED_Scan	831
3.17.8	HAL_LCDLED_GetState	831
3.18	LPTIM 模块	831
3.18.1	HAL_LPTIM_Init	836

3.18.2	HAL_LPTIM_DeInit.....	837
3.18.3	HAL_LPTIM_MspInit.....	837
3.18.4	HAL_LPTIM_MspDeInit.....	837
3.18.5	HAL_LPTIM_PWM_Start.....	838
3.18.6	HAL_LPTIM_PWM_Stop.....	838
3.18.7	HAL_LPTIM_PWM_Start_IT.....	838
3.18.8	HAL_LPTIM_PWM_Stop_IT.....	839
3.18.9	HAL_LPTIM_OnePulse_Start.....	839
3.18.10	HAL_LPTIM_OnePulse_Stop.....	839
3.18.11	HAL_LPTIM_OnePulse_Start_IT.....	840
3.18.12	HAL_LPTIM_OnePulse_Stop_IT.....	840
3.18.13	HAL_LPTIM_SetOnce_Start.....	840
3.18.14	HAL_LPTIM_SetOnce_Stop.....	841
3.18.15	HAL_LPTIM_SetOnce_Start_IT.....	841
3.18.16	HAL_LPTIM_SetOnce_Stop_IT.....	841
3.18.17	HAL_LPTIM_Encoder_Start.....	842
3.18.18	HAL_LPTIM_Encoder_Stop.....	842
3.18.19	HAL_LPTIM_Encoder_Start_IT.....	842
3.18.20	HAL_LPTIM_Encoder_Stop_IT.....	843
3.18.21	HAL_LPTIM_DualEncoder_Start.....	843
3.18.22	HAL_LPTIM_DualEncoder_Stop.....	843
3.18.23	HAL_LPTIM_DualEncoder_Start_IT.....	843
3.18.24	HAL_LPTIM_DualEncoder_Stop_IT.....	844
3.18.25	HAL_LPTIM_TimeOut_Start.....	844
3.18.26	HAL_LPTIM_TimeOut_Stop.....	844
3.18.27	HAL_LPTIM_TimeOut_Start_IT.....	845
3.18.28	HAL_LPTIM_TimeOut_Stop_IT.....	845
3.18.29	HAL_LPTIM_Counter_Start.....	845
3.18.30	HAL_LPTIM_Counter_Stop.....	846
3.18.31	HAL_LPTIM_Counter_Start_IT.....	846
3.18.32	HAL_LPTIM_Counter_Stop_IT.....	846
3.18.33	HAL_LPTIM_ReadCounter.....	847
3.18.34	HAL_LPTIM_ReadAutoReload.....	847
3.18.35	HAL_LPTIM_ReadCompare.....	847
3.18.36	HAL_LPTIM_IRQHandler.....	847
3.18.37	HAL_LPTIM_CompareMatchCallback.....	848
3.18.38	HAL_LPTIM_AutoReloadMatchCallback.....	848
3.18.39	HAL_LPTIM_TriggerCallback.....	848
3.18.40	HAL_LPTIM_CompareWriteCallback.....	849
3.18.41	HAL_LPTIM_AutoReloadWriteCallback.....	849
3.18.42	HAL_LPTIM_DirectionUpCallback.....	849
3.18.43	HAL_LPTIM_DirectionDownCallback.....	849
3.18.44	HAL_LPTIM_DualInput1IdleErrorCallback.....	850
3.18.45	HAL_LPTIM_DualInput2IdleErrorCallback.....	850
3.18.46	HAL_LPTIM_DualB2BErrorCallback.....	850
3.18.47	HAL_LPTIM_DualWaveFormErrorCallback.....	851

3.18.48	HAL_LPTIM_GetState.....	851
3.19	OPAMP 模块.....	851
3.19.1	HAL_OPAMP_Init.....	852
3.19.2	HAL_OPAMP_DeInit.....	853
3.19.3	HAL_OPAMP_MspInit.....	853
3.19.4	HAL_OPAMP_MspDeInit.....	853
3.19.5	HAL_OPAMP_Start.....	853
3.19.6	HAL_OPAMP_Stop.....	854
3.19.7	HAL_OPAMP_SelfCalibrate.....	854
3.19.8	HAL_OPAMP_Lock.....	854
3.19.9	HAL_OPAMP_GetState.....	855
3.19.10	HAL_OPAMPEx_SelfCalibrateAll.....	855
3.20	PLA 模块.....	855
3.20.1	HAL_PLA_Init.....	857
3.20.2	HAL_PLA_DeInit.....	857
3.20.3	HAL_PLA_MspInit.....	857
3.20.4	HAL_PLA_MspDeInit.....	857
3.20.5	HAL_PLA_Start.....	858
3.20.6	HAL_PLA_Stop.....	858
3.20.7	HAL_PLA_IRQHandler.....	858
3.20.8	HAL_PLA_GetOutputLevel.....	859
3.20.9	HAL_PLA_RisingTriggerCallback.....	859
3.20.10	HAL_PLA_FallingTriggerCallback.....	859
3.20.11	HAL_PLA_GetState.....	859
3.20.12	HAL_PLA_GetError.....	860
3.21	RCC 模块.....	860
3.21.1	HAL_RCC_DeInit.....	862
3.21.2	HAL_RCC_OscConfig.....	862
3.21.3	HAL_RCC_ClockConfig.....	862
3.21.4	HAL_RCC_MCOConfig.....	862
3.21.5	HAL_RCC_EnableHSECSS.....	863
3.21.6	HAL_RCC_EnablePLLCS.....	863
3.21.7	HAL_RCC_EnableLSECSS.....	864
3.21.8	HAL_RCC_DisableLSECSS.....	864
3.21.9	HAL_RCC_GetSysClockFreq.....	864
3.21.10	HAL_RCC_GetHCLKFreq.....	864
3.21.11	HAL_RCC_GetPCLK1Freq.....	865
3.21.12	HAL_RCC_GetPCLK2Freq.....	865
3.21.13	HAL_RCC_GetOscConfig.....	865
3.21.14	HAL_RCC_GetClockConfig.....	865
3.21.15	HAL_RCC_NMI_IRQHandler.....	866
3.21.16	HAL_RCC_HSECSSCallback.....	866
3.21.17	HAL_RCC_LSECSSCallback.....	866
3.21.18	HAL_RCC_PLLCSCallback.....	867
3.21.19	HAL_RCCEx_PeriphCLKConfig.....	867

3.21.20	HAL_RCCEx_GetPeriphCLKConfig.....	867
3.21.21	HAL_RCCEx_GetPeriphCLKFreq	868
3.21.22	HAL_RCCEx_EnableLSCO.....	868
3.21.23	HAL_RCCEx_DisableLSCO	868
3.21.24	HAL_RCCEx_EnableRTCLowpower	869
3.21.25	HAL_RCCEx_DisableRTCLowpower	869
3.22	SPI 模块.....	869
3.22.1	HAL_SPI_Init.....	872
3.22.2	HAL_SPI_DeInit	872
3.22.3	HAL_SPI_MspInit.....	872
3.22.4	HAL_SPI_MspDeInit	873
3.22.5	HAL_SPI_Transmit	873
3.22.6	HAL_SPI_Receive.....	873
3.22.7	HAL_SPI_TransmitReceive	874
3.22.8	HAL_SPI_Transmit_IT	874
3.22.9	HAL_SPI_Receive_IT.....	874
3.22.10	HAL_SPI_TransmitReceive_IT.....	875
3.22.11	HAL_SPI_Transmit_DMA.....	875
3.22.12	HAL_SPI_Receive_DMA	875
3.22.13	HAL_SPI_TransmitReceive_DMA	876
3.22.14	HAL_SPI_DMABase.....	876
3.22.15	HAL_SPI_DMAResume	876
3.22.16	HAL_SPI_DMAStop.....	877
3.22.17	HAL_SPI_Abort.....	877
3.22.18	HAL_SPI_Abort_IT	877
3.22.19	HAL_SPI_IRQHandler.....	878
3.22.20	HAL_SPI_TxCpltCallback.....	878
3.22.21	HAL_SPI_RxCpltCallback.....	878
3.22.22	HAL_SPI_TxRxCpltCallback	878
3.22.23	HAL_SPI_TxHalfCpltCallback.....	879
3.22.24	HAL_SPI_RxHalfCpltCallback.....	879
3.22.25	HAL_SPI_TxRxHalfCpltCallback	879
3.22.26	HAL_SPI_ErrorCallback	879
3.22.27	HAL_SPI_AbortCpltCallback	880
3.22.28	HAL_SPI_RxFifoFullCallback.....	880
3.22.29	HAL_SPI_TxFifoEmptyCallback.....	880
3.22.30	HAL_SPI_GetState.....	881
3.22.31	HAL_SPI_GetError	881
3.23	TIM 模块.....	881
3.23.1	HAL_TIM_OC_ConfigChannel	896
3.23.2	HAL_TIM_PWM_ConfigChannel	896
3.23.3	HAL_TIM_IC_ConfigChannel.....	897
3.23.4	HAL_TIM_OnePulse_ConfigChannel	897
3.23.5	HAL_TIM_ConfigOCrefClear	898
3.23.6	HAL_TIM_ConfigClockSource	898

3.23.7	HAL_TIM_ConfigTI1Input.....	898
3.23.8	HAL_TIM_SlaveConfigSynchro.....	899
3.23.9	HAL_TIM_SlaveConfigSynchro_IT.....	899
3.23.10	HAL_TIM_DMABurst_WriteStart.....	899
3.23.11	HAL_TIM_DMABurst_MultiWriteStart.....	900
3.23.12	HAL_TIM_DMABurst_WriteStop.....	901
3.23.13	HAL_TIM_DMABurst_ReadStart.....	901
3.23.14	HAL_TIM_DMABurst_MultiReadStart.....	902
3.23.15	HAL_TIM_DMABurst_ReadStop.....	903
3.23.16	HAL_TIM_GenerateEvent.....	903
3.23.17	HAL_TIM_ReadCapturedValue.....	904
3.23.18	HAL_TIM_PeriodElapsedCallback.....	904
3.23.19	HAL_TIM_PeriodElapsedCallback.....	904
3.23.20	HAL_TIM_OC_DelayElapsedCallback.....	905
3.23.21	HAL_TIM_IC_CaptureCallback.....	905
3.23.22	HAL_TIM_IC_CaptureHalfCpltCallback.....	905
3.23.23	HAL_TIM_PWM_PulseFinishedCallback.....	906
3.23.24	HAL_TIM_PWM_PulseFinishedHalfCpltCallback.....	906
3.23.25	HAL_TIM_TriggerCallback.....	906
3.23.26	HAL_TIM_TriggerHalfCpltCallback.....	906
3.23.27	HAL_TIM_ErrorCallback.....	907
3.23.28	HAL_TIM_Base_GetState.....	907
3.23.29	HAL_TIM_OC_GetState.....	907
3.23.30	HAL_TIM_PWM_GetState.....	908
3.23.31	HAL_TIM_IC_GetState.....	908
3.23.32	HAL_TIM_OnePulse_GetState.....	908
3.23.33	HAL_TIM_Encoder_GetState.....	908
3.23.34	HAL_TIM_GetActiveChannel.....	909
3.23.35	HAL_TIM_GetChannelState.....	909
3.23.36	HAL_TIM_DMABurstState.....	909
3.23.37	HAL_TIM_Base_Init.....	910
3.23.38	HAL_TIM_Base_DeInit.....	910
3.23.39	HAL_TIM_Base_MspInit.....	910
3.23.40	HAL_TIM_Base_MspDeInit.....	910
3.23.41	HAL_TIM_Base_Start.....	911
3.23.42	HAL_TIM_Base_Stop.....	911
3.23.43	HAL_TIM_Base_Start_IT.....	911
3.23.44	HAL_TIM_Base_Stop_IT.....	912
3.23.45	HAL_TIM_Base_Start_DMA.....	912
3.23.46	HAL_TIM_Base_Stop_DMA.....	912
3.23.47	HAL_TIM_OC_Init.....	913
3.23.48	HAL_TIM_OC_DeInit.....	913
3.23.49	HAL_TIM_OC_MspInit.....	913
3.23.50	HAL_TIM_OC_MspDeInit.....	913
3.23.51	HAL_TIM_OC_Start.....	914
3.23.52	HAL_TIM_OC_Stop.....	914

3.23.53	HAL_TIM_OC_Start_IT	914
3.23.54	HAL_TIM_OC_Stop_IT	915
3.23.55	HAL_TIM_OC_Start_DMA.....	915
3.23.56	HAL_TIM_OC_Stop_DMA.....	915
3.23.57	HAL_TIM_PWM_Init.....	916
3.23.58	HAL_TIM_PWM_DeInit	916
3.23.59	HAL_TIM_PWM_MspInit.....	916
3.23.60	HAL_TIM_PWM_MspDeInit	917
3.23.61	HAL_TIM_PWM_Start.....	917
3.23.62	HAL_TIM_PWM_Stop	917
3.23.63	HAL_TIM_PWM_Start_IT	918
3.23.64	HAL_TIM_PWM_Stop_IT	918
3.23.65	HAL_TIM_PWM_Start_DMA.....	918
3.23.66	HAL_TIM_PWM_Stop_DMA.....	919
3.23.67	HAL_TIM_IC_Init	919
3.23.68	HAL_TIM_IC_DeInit.....	919
3.23.69	HAL_TIM_IC_MspInit	920
3.23.70	HAL_TIM_IC_MspDeInit.....	920
3.23.71	HAL_TIM_IC_Start	920
3.23.72	HAL_TIM_IC_Stop.....	921
3.23.73	HAL_TIM_IC_Start_IT.....	921
3.23.74	HAL_TIM_IC_Stop_IT	921
3.23.75	HAL_TIM_IC_Start_DMA	922
3.23.76	HAL_TIM_IC_Stop_DMA	922
3.23.77	HAL_TIM_OnePulse_Init.....	922
3.23.78	HAL_TIM_OnePulse_DeInit	923
3.23.79	HAL_TIM_OnePulse_MspInit.....	923
3.23.80	HAL_TIM_OnePulse_MspDeInit	923
3.23.81	HAL_TIM_OnePulse_Start.....	924
3.23.82	HAL_TIM_OnePulse_Stop	924
3.23.83	HAL_TIM_OnePulse_Start_IT	924
3.23.84	HAL_TIM_OnePulse_Stop_IT.....	924
3.23.85	HAL_TIM_Encoder_Init	925
3.23.86	HAL_TIM_Encoder_DeInit	925
3.23.87	HAL_TIM_Encoder_MspInit.....	925
3.23.88	HAL_TIM_Encoder_MspDeInit	926
3.23.89	HAL_TIM_Encoder_Start.....	926
3.23.90	HAL_TIM_Encoder_Stop	926
3.23.91	HAL_TIM_Encoder_Start_IT	927
3.23.92	HAL_TIM_Encoder_Stop_IT.....	927
3.23.93	HAL_TIM_Encoder_Start_DMA.....	927
3.23.94	HAL_TIM_Encoder_Stop_DMA.....	928
3.23.95	HAL_TIM_IRQHandler	928
3.23.96	HAL_TIMEx_HallSensor_Init	928
3.23.97	HAL_TIMEx_HallSensor_DeInit.....	929
3.23.98	HAL_TIMEx_HallSensor_MspInit	929

3.23.99	HAL_TIMEx_HallSensor_MspDeInit.....	929
3.23.100	HAL_TIMEx_HallSensor_Start.....	929
3.23.101	HAL_TIMEx_HallSensor_Stop.....	930
3.23.102	HAL_TIMEx_HallSensor_Start_IT.....	930
3.23.103	HAL_TIMEx_HallSensor_Stop_IT.....	930
3.23.104	HAL_TIMEx_HallSensor_Start_DMA.....	931
3.23.105	HAL_TIMEx_HallSensor_Stop_DMA.....	931
3.23.106	HAL_TIMEx_OCN_Start.....	931
3.23.107	HAL_TIMEx_OCN_Stop.....	932
3.23.108	HAL_TIMEx_OCN_Start_IT.....	932
3.23.109	HAL_TIMEx_OCN_Stop_IT.....	932
3.23.110	HAL_TIMEx_OCN_Start_DMA.....	933
3.23.111	HAL_TIMEx_OCN_Stop_DMA.....	933
3.23.112	HAL_TIMEx_PWMN_Start.....	933
3.23.113	HAL_TIMEx_PWMN_Stop.....	934
3.23.114	HAL_TIMEx_PWMN_Start_IT.....	934
3.23.115	HAL_TIMEx_PWMN_Stop_IT.....	934
3.23.116	HAL_TIMEx_PWMN_Start_DMA.....	935
3.23.117	HAL_TIMEx_PWMN_Stop_DMA.....	935
3.23.118	HAL_TIMEx_OnePulseN_Start.....	935
3.23.119	HAL_TIMEx_OnePulseN_Stop.....	936
3.23.120	HAL_TIMEx_OnePulseN_Start_IT.....	936
3.23.121	HAL_TIMEx_OnePulseN_Stop_IT.....	936
3.23.122	HAL_TIMEx_ConfigCommutEvent.....	937
3.23.123	HAL_TIMEx_ConfigCommutEvent_IT.....	937
3.23.124	HAL_TIMEx_ConfigCommutEvent_DMA.....	938
3.23.125	HAL_TIMEx_MasterConfigSynchronization.....	938
3.23.126	HAL_TIMEx_ConfigBreakDeadTime.....	938
3.23.127	HAL_TIMEx_ConfigBreakInput.....	939
3.23.128	HAL_TIMEx_GroupChannel5.....	939
3.23.129	HAL_TIMEx_RemapConfig.....	939
3.23.130	HAL_TIMEx_TISelection.....	940
3.23.131	HAL_TIMEx_DisarmBreakInput.....	940
3.23.132	HAL_TIMEx_ReArmBreakInput.....	941
3.23.133	HAL_TIMEx_CommutCallback.....	941
3.23.134	HAL_TIMEx_CommutHalfCpltCallback.....	941
3.23.135	HAL_TIMEx_BreakCallback.....	941
3.23.136	HAL_TIMEx_BreakCallback2.....	942
3.23.137	HAL_TIMEx_HallSensor_GetState.....	942
3.23.138	HAL_TIMEx_GetChannelNState.....	942
3.24	UART 模块.....	943
3.24.1	HAL_UART_Init.....	950
3.24.2	HAL_HalfDuplex_Init.....	951
3.24.3	HAL_MultiProcessor_Init.....	951
3.24.4	HAL_UART_DeInit.....	951

3.24.5	HAL_UART_MspInit.....	952
3.24.6	HAL_UART_MspDeInit	952
3.24.7	HAL_UART_Transmit	952
3.24.8	HAL_UART_Receive.....	952
3.24.9	HAL_UART_Transmit_IT.....	953
3.24.10	HAL_UART_Receive_IT	953
3.24.11	HAL_UART_Transmit_DMA	953
3.24.12	HAL_UART_Receive_DMA	954
3.24.13	HAL_UART_DMABase	954
3.24.14	HAL_UART_DMAResume	954
3.24.15	HAL_UART_DMAStop	955
3.24.16	HAL_UART_Abort	955
3.24.17	HAL_UART_AbortTransmit	955
3.24.18	HAL_UART_AbortReceive	956
3.24.19	HAL_UART_Abort_IT.....	956
3.24.20	HAL_UART_AbortTransmit_IT	956
3.24.21	HAL_UART_AbortReceive_IT.....	956
3.24.22	HAL_UART_IRQHandler.....	957
3.24.23	HAL_UART_TxHalfCpltCallback	957
3.24.24	HAL_UART_TxCpltCallback	957
3.24.25	HAL_UART_RxHalfCpltCallback	958
3.24.26	HAL_UART_RxCpltCallback	958
3.24.27	HAL_UART_ErrorCallback	958
3.24.28	HAL_UART_AbortCpltCallback	958
3.24.29	HAL_UART_AbortTransmitCpltCallback	959
3.24.30	HAL_UART_AbortReceiveCpltCallback	959
3.24.31	HAL_UART_ReceiverTimeout_Config	959
3.24.32	HAL_UART_EnableReceiverTimeout	960
3.24.33	HAL_UART_DisableReceiverTimeout	960
3.24.34	HAL_MultiProcessor_EnableMuteMode	960
3.24.35	HAL_MultiProcessor_DisableMuteMode	960
3.24.36	HAL_MultiProcessor_EnterMuteMode	961
3.24.37	HAL_HalfDuplex_EnableTransmitter	961
3.24.38	HAL_HalfDuplex_EnableReceiver	961
3.24.39	HAL_UART_GetState	962
3.24.40	HAL_UART_GetError	962
3.24.41	HAL_RS485Ex_Init	962
3.24.42	HAL_UARTEx_WakeupCallback	963
3.24.43	HAL_UARTEx_RxFifoFullCallback	963
3.24.44	HAL_UARTEx_TxFifoEmptyCallback	963
3.24.45	HAL_UARTEx_RxEventCallback	963
3.24.46	HAL_UARTEx_StopModeWakeUpSourceConfig.....	964
3.24.47	HAL_UARTEx_EnableStopMode.....	964
3.24.48	HAL_UARTEx_DisableStopMode	964
3.24.49	HAL_MultiProcessorEx_AddressLength_Set	965
3.24.50	HAL_UARTEx_EnableFifoMode	965

3.24.51	HAL_UARTEx_DisableFifoMode	965
3.24.52	HAL_UARTEx_SetTxFifoThreshold	966
3.24.53	HAL_UARTEx_SetRxFifoThreshold	966
3.24.54	HAL_UARTEx_ReceiveToIdle	966
3.24.55	HAL_UARTEx_ReceiveToIdle_IT	967
3.24.56	HAL_UARTEx_ReceiveToIdle_DMA	967
3.25	USART 模块	967
3.25.1	HAL_USART_Init	973
3.25.2	HAL_USART_DeInit	973
3.25.3	HAL_USART_MspInit	973
3.25.4	HAL_USART_MspDeInit	973
3.25.5	HAL_USART_Transmit	974
3.25.6	HAL_USART_Receive	974
3.25.7	HAL_USART_TransmitReceive	974
3.25.8	HAL_USART_Transmit_IT	975
3.25.9	HAL_USART_Receive_IT	975
3.25.10	HAL_USART_TransmitReceive_IT	975
3.25.11	HAL_USART_Transmit_DMA	976
3.25.12	HAL_USART_Receive_DMA	976
3.25.13	HAL_USART_TransmitReceive_DMA	977
3.25.14	HAL_USART_DMABase	977
3.25.15	HAL_USART_DMAResume	977
3.25.16	HAL_USART_DMAStop	978
3.25.17	HAL_USART_Abort	978
3.25.18	HAL_USART_Abort_IT	978
3.25.19	HAL_USART_IRQHandler	978
3.25.20	HAL_USART_TxHalfCpltCallback	979
3.25.21	HAL_USART_TxCpltCallback	979
3.25.22	HAL_USART_RxHalfCpltCallback	979
3.25.23	HAL_USART_RxCpltCallback	980
3.25.24	HAL_USART_TxRxCpltCallback	980
3.25.25	HAL_USART_ErrorCallback	980
3.25.26	HAL_USART_AbortCpltCallback	980
3.25.27	HAL_USART_GetState	981
3.25.28	HAL_USART_GetError	981
3.25.29	HAL_USARTEx_RxFifoFullCallback	981
3.25.30	HAL_USARTEx_TxFifoEmptyCallback	982
3.25.31	HAL_USARTEx_EnableSlaveMode	982
3.25.32	HAL_USARTEx_DisableSlaveMode	982
3.25.33	HAL_USARTEx_ConfigNSS	982
3.25.34	HAL_USARTEx_EnableFifoMode	983
3.25.35	HAL_USARTEx_DisableFifoMode	983
3.25.36	HAL_USARTEx_SetTxFifoThreshold	983
3.25.37	HAL_USARTEx_SetRxFifoThreshold	984
3.26	WWDG 模块	984

3.26.1	HAL_WWDG_Init	985
3.26.2	HAL_WWDG_MspInit	985
3.26.3	HAL_WWDG_Refresh	985
3.26.4	HAL_WWDG_IRQHandler	986
3.26.5	HAL_WWDG_EarlyWakeupCallback	986
3.27	CORDIC 模块	986
3.27.1	HAL_CORDIC_Init	988
3.27.2	HAL_CORDIC_DeInit	988
3.27.3	HAL_CORDIC_MspInit	988
3.27.4	HAL_CORDIC_MspDeInit	988
3.27.5	HAL_CORDIC_Configure	989
3.27.6	HAL_CORDIC_Calculate	989
3.27.7	HAL_CORDIC_CalculateZO	989
3.27.8	HAL_CORDIC_Calculate_IT	990
3.27.9	HAL_CORDIC_Calculate_DMA	990
3.27.10	HAL_CORDIC_ErrorCallback	991
3.27.11	HAL_CORDIC_CalculateCpltCallback	991
3.27.12	HAL_CORDIC_IRQHandler	991
3.27.13	HAL_CORDIC_GetState	991
3.27.14	HAL_CORDIC_GetError	992

1 概述

本文列举了 TM32G05x 系列 API 函数，包括基础 LL 层和基础 HAL 层。

2 API（基础 LL 层）

宏定义	数值	含义
ErrorStatus		
SUCCESS	0x00	成功
ERROR	!0x00	错误
FunctionalState		
DISABLE	0x00	禁止使能
ENABLE	!0x00	使能

2.1 IWDG 模块

IWDG 可检测并解决芯片发生的异常，在计数器达到超时值时触发系统复位。该 API 模块提供对模块的控制功能，配置预分频、重载、计数窗口的功能，以及获取模块状态的功能。

属性	类型	字段名	含义
IWDG_TypeDef			
只写	uint32_t	KR	控制寄存器
读写	uint32_t	PR	预分频寄存器
读写	uint32_t	RLR	重载寄存器
只读	uint32_t	SR	状态寄存器
读写	uint32_t	WINR	窗口寄存器

2.1.1 LL_IWDG_Enable

2.1.1.1 功能介绍

开启独立看门狗。

2.1.1.2 接口定义

函数接口	void LL_IWDG_Enable (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	无
资源使用	
说明	

2.1.2 LL_IWDG_ReloadCounter

2.1.2.1 功能介绍

重新加载 IWDG 计数器来重新加载寄存器中定义的值。

2.1.2.2 接口定义

函数接口	void LL_IWDG_ReloadCounter (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	无
资源使用	
说明	

2.1.3 LL_IWDG_EnableWriteAccess

2.1.3.1 功能介绍

开启对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的写访问。

2.1.3.2 接口定义

函数接口	void LL_IWDG_EnableWriteAccess (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	无
资源使用	
说明	

2.1.4 LL_IWDG_DisableWriteAccess

2.1.4.1 功能介绍

关闭对 IWDG_PR、IWDG_RLR 和 IWDG_WINR 寄存器的写访问。

2.1.4.2 接口定义

函数接口	void LL_IWDG_DisableWriteAccess (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	无
资源使用	
说明	

2.1.5 LL_IWDG_SetPrescaler

2.1.5.1 功能介绍

设置 IWDG 的预分频值。

2.1.5.2 接口定义

函数接口	void LL_IWDG_SetPrescaler (IWDG_TypeDef * IWDGx, uint32_t Prescaler)
输入	IWDG_TypeDef * IWDGx uint32_t Prescaler: {LL_IWDG_PRESCALER_4, LL_IWDG_PRESCALER_8, LL_IWDG_PRESCALER_16, LL_IWDG_PRESCALER_32, LL_IWDG_PRESCALER_64, LL_IWDG_PRESCALER_128,

	LL_IWDG_PRESCALER_256}
输出	无
返回值	无
资源使用	
说明	

2.1.6 LL_IWDG_GetPrescaler

2.1.6.1 功能介绍

获取 IWDG 的预分频值。

2.1.6.2 接口定义

函数接口	uint32_t LL_IWDG_GetPrescaler (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	{LL_IWDG_PRESCALER_4, LL_IWDG_PRESCALER_8, LL_IWDG_PRESCALER_16, LL_IWDG_PRESCALER_32, LL_IWDG_PRESCALER_64, LL_IWDG_PRESCALER_128, LL_IWDG_PRESCALER_256}
资源使用	
说明	

2.1.7 LL_IWDG_SetReloadCounter

2.1.7.1 功能介绍

设置 IWDG 的重载值。

2.1.7.2 接口定义

函数接口	void LL_IWDG_SetReloadCounter (IWDG_TypeDef * IWDGx, uint32_t Counter)
输入	IWDG_TypeDef * IWDGx uint32_t Counter : [0,0x0FFF]
输出	无
返回值	无
资源使用	
说明	

2.1.8 LL_IWDG_GetReloadCounter

2.1.8.1 功能介绍

获取 IWDG 的重载值。

2.1.8.2 接口定义

函数接口	uint32_t LL_IWDG_GetReloadCounter (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	[0,0x0FFF]

资源使用	
说明	

2.1.9 LL_IWDG_SetWindow

2.1.9.1 功能介绍

设置 IWDG 的窗口值。

2.1.9.2 接口定义

函数接口	void LL_IWDG_SetWindow (IWDG_TypeDef * IWDGx, uint32_t Window)
输入	IWDG_TypeDef * IWDGx uint32_t Window: [0,0xFFFF]
输出	无
返回值	无
资源使用	
说明	

2.1.10 LL_IWDG_GetWindow

2.1.10.1 功能介绍

获取 IWDG 的窗口值。

2.1.10.2 接口定义

函数接口	uint32_t LL_IWDG_GetWindow (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	[0,0xFFFF]
资源使用	
说明	

2.1.11 LL_IWDG_IsActiveFlag_PVU

2.1.11.1 功能介绍

检查看门狗计数器预分频值是否正在更新。

2.1.11.2 接口定义

函数接口	uint32_t LL_IWDG_IsActiveFlag_PVU (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	{0,1}
资源使用	
说明	

2.1.12 LL_IWDG_IsActiveFlag_RVU

2.1.12.1 功能介绍

检查看门狗计数器重载值是否正在更新。

2.1.12.2 接口定义

函数接口	uint32_t LL_IWDG_IsActiveFlag_RVU (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	{0,1}
资源使用	
说明	

2.1.13 LL_IWDG_IsActiveFlag_WVU

2.1.13.1 功能介绍

检查看门狗计数器窗口值是否正在更新。

2.1.13.2 接口定义

函数接口	uint32_t LL_IWDG_IsActiveFlag_WVU (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	{0,1}
资源使用	
说明	

2.1.14 LL_IWDG_IsActiveFlag

2.1.14.1 功能介绍

检查标志位是否设置。

2.1.14.2 接口定义

函数接口	uint32_t LL_IWDG_IsActiveFlag (IWDG_TypeDef * IWDGx, uint32_t Flag)
输入	IWDG_TypeDef * IWDGx uint32_t Flag: {LL_IWDG_SR_PVU, LL_IWDG_SR_RVU, LL_IWDG_SR_WVU}
输出	无
返回值	{0,1}
资源使用	
说明	

2.1.15 LL_IWDG_IsReady

2.1.15.1 功能介绍

检查预分频值、重载值、窗口值是否可以更新。

2.1.15.2 接口定义

函数接口	uint32_t LL_IWDG_IsReady (IWDG_TypeDef * IWDGx)
输入	IWDG_TypeDef * IWDGx
输出	无
返回值	{0,1}

资源使用	
说明	

2.2 CORTEX 模块

2.2.1 LL_SYSTICK_IsActiveCounterFlag

2.2.1.1 功能介绍

检查 SysTick 计数器标志是否激活。

2.2.1.2 接口定义

函数接口	uint32_t LL_SYSTICK_IsActiveCounterFlag (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.2.2 LL_SYSTICK_SetClkSource

2.2.2.1 功能介绍

配置 SysTick 时钟源。

2.2.2.2 接口定义

函数接口	void LL_SYSTICK_SetClkSource (uint32_t Source)
输入	uint32_t Source: { LL_SYSTICK_CLKSOURCE_HCLK_DIV8 , LL_SYSTICK_CLKSOURCE_HCLK }
输出	无
返回值	无
资源使用	
说明	

2.2.3 LL_SYSTICK_GetClkSource

2.2.3.1 功能介绍

获取 SysTick 时钟源。

2.2.3.2 接口定义

函数接口	uint32_t LL_SYSTICK_GetClkSource (void)
输入	无
输出	无
返回值	{ LL_SYSTICK_CLKSOURCE_HCLK_DIV8 , LL_SYSTICK_CLKSOURCE_HCLK }
资源使用	
说明	

2.2.4 LL_SYSTICK_EnableIT

2.2.4.1 功能介绍

使能 SysTick 异常请求。

2.2.4.2 接口定义

函数接口	void LL_SYSTICK_EnableIT (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.5 LL_SYSTICK_DisableIT

2.2.5.1 功能介绍

禁用 SysTick 异常请求。

2.2.5.2 接口定义

函数接口	void LL_SYSTICK_DisableIT (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.6 LL_SYSTICK_IsEnabledIT

2.2.6.1 功能介绍

检查 SYSTICK 中断是否使能或禁用。

2.2.6.2 接口定义

函数接口	uint32_t LL_SYSTICK_IsEnabledIT (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.2.7 LL_LPM_EnableSleep

2.2.7.1 功能介绍

处理器采用睡眠模式作为低功耗模式。

2.2.7.2 接口定义

函数接口	void LL_LPM_EnableSleep (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.2.8 LL_LPM_EnableDeepSleep

2.2.8.1 功能介绍

处理器采用深度睡眠模式作为低功耗模式。

2.2.8.2 接口定义

函数接口	void LL_LPM_EnableDeepSleep (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.9 LL_LPM_EnableSleepOnExit

2.2.9.1 功能介绍

退出最低优先级中断服务函数时进入低功耗模式。

2.2.9.2 接口定义

函数接口	void LL_LPM_EnableSleepOnExit (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.10 LL_LPM_DisableSleepOnExit

2.2.10.1 功能介绍

退出最低优先级中断服务函数时不进入低功耗模式。

2.2.10.2 接口定义

函数接口	void LL_LPM_DisableSleepOnExit (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.11 LL_LPM_IsEnabledSleepOnExit

2.2.11.1 功能介绍

退出最低优先级中断服务函数时是否进入低功耗模式。

2.2.11.2 接口定义

函数接口	uint32_t LL_LPM_IsEnabledSleepOnExit (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.2.12 LL_LPM_EnableEventOnPend

2.2.12.1 功能介绍

启用事件和所有中断,包括禁用中断,可以唤醒处理器。

2.2.12.2 接口定义

函数接口	void LL_LPM_EnableEventOnPend (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.13 LL_LPM_DisableEventOnPend

2.2.13.1 功能介绍

只有启用的中断或事件才能唤醒处理器，禁用的中断将被排除。

2.2.13.2 接口定义

函数接口	void LL_LPM_DisableEventOnPend (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.2.14 LL_CPUID_GetImplementer

2.2.14.1 功能介绍

获取实施者的代码。

2.2.14.2 接口定义

函数接口	uint32_t LL_CPUID_GetImplementer (void)
输入	无
输出	无
返回值	0x41 的 ARM
资源使用	
说明	

2.2.15 LL_CPUID_GetVariant

2.2.15.1 功能介绍

获取变量号。

2.2.15.2 接口定义

函数接口	uint32_t LL_CPUID_GetVariant (void)
输入	无
输出	无
返回值	[0, 255]
资源使用	
说明	

2.2.16 LL_CPUID_GetArchitecture

2.2.16.1 功能介绍

获取架构号。

2.2.16.2 接口定义

函数接口	uint32_t LL_CPUID_GetArchitecture (void)
输入	无
输出	无
返回值	对于 Cortex-M0+设备应该等于 0xC
资源使用	
说明	

2.2.17 LL_CPUID_GetParNo

2.2.17.1 功能介绍

获取零件编号。

2.2.17.2 接口定义

函数接口	uint32_t LL_CPUID_GetParNo (void)
输入	无
输出	无
返回值	对于 Cortex-M0+应该等于 0xC60
资源使用	
说明	

2.2.18 LL_CPUID_GetRevision

2.2.18.1 功能介绍

获取版本号。

2.2.18.2 接口定义

函数接口	uint32_t LL_CPUID_GetRevision (void)
输入	无
输出	无

返回值	[0, 255]
资源使用	
说明	

2.3 EXTI 模块

属性	类型	字段名	含义
LL_EXTI_InitTypeDef			
读写	uint32_t	Line	为范围 0 到 25 的 lines 指定要启用或禁用的 EXTI 行
读写	uint8_t	Mode	指定 EXTI 行的模式
读写	uint8_t	Trigger	为 EXTI 行指定触发沿

2.3.1 LL_EXTI_EnableIT

2.3.1.1 功能介绍

使能 ExtiLine 中断请求。

2.3.1.2 接口定义

函数接口	void LL_EXTI_EnableIT (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0, LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	无
资源使用	
说明	

2.3.2 LL_EXTI_DisableIT

2.3.2.1 功能介绍

禁止使能 ExtiLine 中断请求。

2.3.2.2 接口定义

函数接口	void LL_EXTI_DisableIT (uint32_t ExtiLine)
------	--

输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	无
资源使用	
说明	

2.3.3 LL_EXTI_IsEnabledIT

2.3.3.1 功能介绍

检查是否启用了 ExtiLine 中断请求。

2.3.3.2 接口定义

函数接口	uint32_t LL_EXTI_IsEnabledIT (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.4 LL_EXTI_EnableEvent

2.3.4.1 功能介绍

使能 ExtiLine 事件请求。

2.3.4.2 接口定义

函数接口	void LL_EXTI_EnableEvent (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5,

	LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	无
资源使用	
说明	

2.3.5 LL_EXTI_DisableEvent

2.3.5.1 功能介绍

禁止使能 ExtiLine 事件请求。

2.3.5.2 接口定义

函数接口	void LL_EXTI_DisableEvent (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	无
资源使用	
说明	

2.3.6 LL_EXTI_IsEnabledEvent

2.3.6.1 功能介绍

检查是否启用了 ExtiLine 事件请求。

2.3.6.2 接口定义

函数接口	uint32_t LL_EXTI_IsEnabledEvent (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14,

	LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.7 LL_EXTI_EnableRisingTrig

2.3.7.1 功能介绍

使能上升沿触发。

2.3.7.2 接口定义

函数接口	void LL_EXTI_EnableRisingTrig (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	无
资源使用	
说明	

2.3.8 LL_EXTI_DisableRisingTrig

2.3.8.1 功能介绍

禁止上升沿触发。

2.3.8.2 接口定义

函数接口	void LL_EXTI_DisableRisingTrig (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	无
资源使用	
说明	

2.3.9 LL_EXTI_IsEnabledRisingTrig

2.3.9.1 功能介绍

检查是否使能上升沿触发。

2.3.9.2 接口定义

函数接口	uint32_t LL_EXTI_IsEnabledRisingTrig (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.10 LL_EXTI_EnableFallingTrig

2.3.10.1 功能介绍

使能下降沿触发。

2.3.10.2 接口定义

函数接口	void LL_EXTI_EnableFallingTrig (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	无
资源使用	
说明	

2.3.11 LL_EXTI_DisableFallingTrig

2.3.11.1 功能介绍

禁止下降沿触发。

2.3.11.2 接口定义

函数接口	void LL_EXTI_DisableFallingTrig (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5,

	LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	无
资源使用	
说明	

2.3.12 LL_EXTI_IsEnabledFallingTrig

2.3.12.1 功能介绍

检查是否使能下降沿触发。

2.3.12.2 接口定义

函数接口	uint32_t LL_EXTI_IsEnabledFallingTrig (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.13 LL_EXTI_IsActiveFallingFlag

2.3.13.1 功能介绍

检查下降沿标志是否设置。

2.3.13.2 接口定义

函数接口	uint32_t LL_EXTI_IsActiveFallingFlag (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.14 LL_EXTI_ReadFallingFlag

2.3.14.1 功能介绍

读取下降沿标志。

2.3.14.2 接口定义

函数接口	uint32_t LL_EXTI_ReadFallingFlag (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.15 LL_EXTI_ClearFallingFlag

2.3.15.1 功能介绍

清除下降沿标志。

2.3.15.2 接口定义

函数接口	void LL_EXTI_ClearFallingFlag (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	无
资源使用	
说明	

2.3.16 LL_EXTI_IsActiveRisingFlag

2.3.16.1 功能介绍

检查上升沿标志是否设置。

2.3.16.2 接口定义

函数接口	uint32_t LL_EXTI_IsActiveRisingFlag (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5,

	LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.17 LL_EXTI_ReadRisingFlag

2.3.17.1 功能介绍

读取上升沿标志。

2.3.17.2 接口定义

函数接口	uint32_t LL_EXTI_ReadRisingFlag (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	{0, 1}
资源使用	
说明	

2.3.18 LL_EXTI_ClearRisingFlag

2.3.18.1 功能介绍

读取上升沿标志。

2.3.18.2 接口定义

函数接口	void_t LL_EXTI_ClearRisingFlag (uint32_t ExtiLine)
输入	uint32_t ExtiLine: { LL_EXTI_LINE_0 , LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11, LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17}
输出	无
返回值	无
资源使用	
说明	

2.3.19 LL_EXTI_SetEXTISource_0_7

2.3.19.1 功能介绍

配置 EXTI 外部中断源输入。

2.3.19.2 接口定义

函数接口	void LL_EXTI_SetEXTISource_0_7 (uint32_t Port, uint32_t Line)
输入	uint32_t Port: { LL_EXTI_CONFIG_PORTA, LL_EXTI_CONFIG_PORTB, LL_EXTI_CONFIG_PORTC, LL_EXTI_CONFIG_PORTD} uint32_t Line: { LL_EXTI_CONFIG_LINE0, LL_EXTI_CONFIG_LINE1, LL_EXTI_CONFIG_LINE2, LL_EXTI_CONFIG_LINE3, LL_EXTI_CONFIG_LINE4, LL_EXTI_CONFIG_LINE5, LL_EXTI_CONFIG_LINE6, LL_EXTI_CONFIG_LINE7}
输出	无
返回值	无
资源使用	
说明	

2.3.20 LL_EXTI_GetEXTISource_0_7

2.3.20.1 功能介绍

获取 EXTI 外部中断源输入。

2.3.20.2 接口定义

函数接口	uint32_t LL_EXTI_GetEXTISource_0_7 (uint32_t Line)
输入	uint32_t Line: { LL_EXTI_CONFIG_LINE0, LL_EXTI_CONFIG_LINE1, LL_EXTI_CONFIG_LINE2, LL_EXTI_CONFIG_LINE3, LL_EXTI_CONFIG_LINE4, LL_EXTI_CONFIG_LINE5, LL_EXTI_CONFIG_LINE6, LL_EXTI_CONFIG_LINE7}
输出	无
返回值	{ LL_EXTI_CONFIG_PORTA, LL_EXTI_CONFIG_PORTB, LL_EXTI_CONFIG_PORTC, LL_EXTI_CONFIG_PORTD}
资源使用	
说明	

2.3.21 LL_EXTI_SetEXTISource_8_15

2.3.21.1 功能介绍

配置 EXTI 外部中断源输入。

2.3.21.2 接口定义

函数接口	void LL_EXTI_SetEXTISource_8_15 (uint32_t Port, uint32_t Line)
输入	uint32_t Port: { LL_EXTI_CONFIG_PORTA, LL_EXTI_CONFIG_PORTB,

	LL_EXTI_CONFIG_PORTC, LL_EXTI_CONFIG_PORTD} uint32_t Line: { LL_EXTI_CONFIG_LINE8, LL_EXTI_CONFIG_LINE9, LL_EXTI_CONFIG_LINE10, LL_EXTI_CONFIG_LINE11, LL_EXTI_CONFIG_LINE12, LL_EXTI_CONFIG_LINE13, LL_EXTI_CONFIG_LINE14, LL_EXTI_CONFIG_LINE15}
输出	无
返回值	无
资源使用	
说明	

2.3.22 LL_EXTI_GetEXTISource_8_15

2.3.22.1 功能介绍

获取 EXTI 外部中断源输入。

2.3.22.2 接口定义

函数接口	uint32 LL_EXTI_GetEXTISource_8_15 (uint32_t Line)
输入	uint32_t Line: { LL_EXTI_CONFIG_LINE8, LL_EXTI_CONFIG_LINE9, LL_EXTI_CONFIG_LINE10, LL_EXTI_CONFIG_LINE11, LL_EXTI_CONFIG_LINE12, LL_EXTI_CONFIG_LINE13, LL_EXTI_CONFIG_LINE14, LL_EXTI_CONFIG_LINE15}
输出	无
返回值	{ LL_EXTI_CONFIG_PORTA, LL_EXTI_CONFIG_PORTB, LL_EXTI_CONFIG_PORTC, LL_EXTI_CONFIG_PORTD}
资源使用	
说明	

2.3.23 LL_EXTI_Init

2.3.23.1 功能介绍

初始化 EXTI。

2.3.23.2 接口定义

函数接口	uint32_t LL_EXTI_Init (LL_EXTI_InitTypeDef * EXTI_InitStruct)
输入	LL_EXTI_InitTypeDef * EXTI_InitStruct
输出	无
返回值	{0x00, 0x01, 0x02}
资源使用	
说明	

2.3.24 LL_EXTI_DeInit

2.3.24.1 功能介绍

清除初始化 EXTI 配置。

2.3.24.2 接口定义

函数接口	uint32_t LL_EXTI_DeInit (void)
输入	无
输出	无
返回值	{0x00, 0x01, 0x02}
资源使用	
说明	

2.3.25 LL_EXTI_StructInit

2.3.25.1 功能介绍

EXTI 结构体初始化。

2.3.25.2 接口定义

函数接口	void LL_EXTI_StructInit (LL_EXTI_InitTypeDef * EXTI_InitStruct)
输入	LL_EXTI_InitTypeDef * EXTI_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.4 FLASH 模块

2.4.1 LL_FLASH_SetCSCConfig

2.4.1.1 功能介绍

设置 FLASH 片选建立时间。

2.4.1.2 接口定义

函数接口	void LL_FLASH_SetCSCConfig(uint32_t CSCConfig)
输入	uint32_t CSCConfig: { LL_FLASH_CS_1CYC, LL_FLASH_CS_2CYC, LL_FLASH_CS_3CYC, LL_FLASH_CS_4CYC, LL_FLASH_CS_5CYC, LL_FLASH_CS_6CYC, LL_FLASH_CS_7CYC}
输出	无
返回值	无
资源使用	
说明	

2.4.2 LL_FLASH_GetCSCConfig

2.4.2.1 功能介绍

获取 FLASH 片选建立时间。

2.4.2.2 接口定义

函数接口	uint32_t LL_FLASH_GetCSCConfig(void)
------	--------------------------------------

输入	无
输出	无
返回值	{LL_FLASH_CS_1CYC, LL_FLASH_CS_2CYC, LL_FLASH_CS_3CYC, LL_FLASH_CS_4CYC, LL_FLASH_CS_5CYC, LL_FLASH_CS_6CYC, LL_FLASH_CS_7CYC}
资源使用	
说明	

2.4.3 LL_FLASH_SetLatency

2.4.3.1 功能介绍

设置 FLASH 读取访问等待周期。

2.4.3.2 接口定义

函数接口	void LL_FLASH_SetLatency (uint32_t Latency)
输入	uint32_t Latency: { LL_FLASH_LATENCY_0, LL_FLASH_LATENCY_1, LL_FLASH_LATENCY_2}
输出	无
返回值	无
资源使用	
说明	

2.4.4 LL_FLASH_GetLatency

2.4.4.1 功能介绍

获取 FLASH 读取访问等待周期。

2.4.4.2 接口定义

函数接口	uint32_t LL_FLASH_GetLatency (void)
输入	无
输出	无
返回值	{ LL_FLASH_LATENCY_0, LL_FLASH_LATENCY_1, LL_FLASH_LATENCY_2}
资源使用	
说明	

2.4.5 LL_FLASH_EnablePrefetch

2.4.5.1 功能介绍

使能 FLASH 指令预取。

2.4.5.2 接口定义

函数接口	void LL_FLASH_EnablePrefetch (void)
输入	无
输出	无
返回值	无

资源使用	
说明	

2.4.6 LL_FLASH_DisablePrefetch

2.4.6.1 功能介绍

禁止 FLASH 指令预取。

2.4.6.2 接口定义

函数接口	void LL_FLASH_DisablePrefetch (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.7 LL_FLASH_IsEnabledPrefetch

2.4.7.1 功能介绍

检查 FLASH 指令预取是否使能。

2.4.7.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledPrefetch (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.8 LL_FLASH_EnableProgRror

2.4.8.1 功能介绍

开启编程错误检查。

2.4.8.2 接口定义

函数接口	void LL_FLASH_EnableProgRror(void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.9 LL_FLASH_DisableProgRror

2.4.9.1 功能介绍

关闭编程错误检查。

2.4.9.2 接口定义

函数接口	void LL_FLASH_DisableProgRror(void)
------	-------------------------------------

输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.10 LL_FLASH_IsEnabledProgRror

2.4.10.1 功能介绍

判断编程错误检查是否开启。

2.4.10.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledProgRror(void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.11 LL_FLASH_IsActiveFlag

2.4.11.1 功能介绍

检查 FLASH 指令预取是否使能。

2.4.11.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag (uint32_t FlagType)
输入	uint32_t FlagType: { LL_FLASH_FLAG_ACE, LL_FLASH_FLAG_RDE, LL_FLASH_FLAG_OPE, LL_FLASH_FLAG_EOP, LL_FLASH_FLAG_BSY, LL_FLASH_FLAG_SIZEE, LL_FLASH_FLAG_PESE, LL_FLASH_FLAG_PGAE, LL_FLASH_FLAG_WRPE, LL_FLASH_FLAG_PROE, LL_FLASH_FLAG_ALL_ERRORS }
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.12 LL_FLASH_GetActiveFlag

2.4.12.1 功能介绍

获取 FLASH 标志位。

2.4.12.2 接口定义

函数接口	uint32_t LL_FLASH_GetActiveFlag (uint32_t FlagType)
输入	uint32_t FlagType: { LL_FLASH_FLAG_ACE, LL_FLASH_FLAG_RDE, LL_FLASH_FLAG_OPE, LL_FLASH_FLAG_EOP,

	LL_FLASH_FLAG_BSY, LL_FLASH_FLAG_SIZEE, LL_FLASH_FLAG_PESE, LL_FLASH_FLAG_PGAE, LL_FLASH_FLAG_WRPE, LL_FLASH_FLAG_PROE, LL_FLASH_FLAG_ALL_ERRORS }
输出	无
返回值	{ LL_FLASH_FLAG_ACE, LL_FLASH_FLAG_RDE, LL_FLASH_FLAG_OPE, LL_FLASH_FLAG_EOP, LL_FLASH_FLAG_BSY, LL_FLASH_FLAG_SIZEE, LL_FLASH_FLAG_PESE, LL_FLASH_FLAG_PGAE, LL_FLASH_FLAG_WRPE, LL_FLASH_FLAG_PROE, LL_FLASH_FLAG_ALL_ERRORS }
资源使用	
说明	

2.4.13 LL_FLASH_ClearFlag

2.4.13.1 功能介绍

清除 FLASH 标志位。

2.4.13.2 接口定义

函数接口	void LL_FLASH_ClearFlag (uint32_t FlagType)
输入	uint32_t FlagType: { LL_FLASH_FLAG_ACE, LL_FLASH_FLAG_RDE, LL_FLASH_FLAG_OPE, LL_FLASH_FLAG_EOP, LL_FLASH_FLAG_BSY, LL_FLASH_FLAG_SIZEE, LL_FLASH_FLAG_PESE, LL_FLASH_FLAG_PGAE, LL_FLASH_FLAG_WRPE, LL_FLASH_FLAG_PROE, LL_FLASH_FLAG_ALL_ERRORS }
输出	无
返回值	无
资源使用	
说明	

2.4.14 LL_FLASH_IsActiveFlag_AccessError

2.4.14.1 功能介绍

检查访问权限错误标志是否使能。

2.4.14.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_AccessError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.15 LL_FLASH_ClearFlag_AccessError

2.4.15.1 功能介绍

清除访问权限错误标志。

2.4.15.2 接口定义

函数接口	void LL_FLASH_ClearFlag_AccessError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.16 LL_FLASH_IsActiveFlag_ReadError

2.4.16.1 功能介绍

检查 PCROP 读取错误标志是否使能。

2.4.16.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ReadError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.17 LL_FLASH_ClearFlag_ReadError

2.4.17.1 功能介绍

清除 PCROP 读取错误标志。

2.4.17.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ReadError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.18 LL_FLASH_IsActiveFlag_ProgEraseInterrupt

2.4.18.1 功能介绍

检查编程/擦除操作异常中断标志是否使能。

2.4.18.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgEraseInterrupt (void)
输入	无
输出	无

返回值	{0, 1}
资源使用	
说明	

2.4.19 LL_FLASH_ClearFlag_ProgEraseInterrupt

2.4.19.1 功能介绍

清除编程/擦除操作异常中断标志。

2.4.19.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgEraseInterrupt (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.20 LL_FLASH_IsActiveFlag_ProgFinished

2.4.20.1 功能介绍

检查编程/擦除操作完成中断标志是否使能。

2.4.20.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgFinished (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.21 LL_FLASH_ClearFlag_ProgFinished

2.4.21.1 功能介绍

清除编程/擦除操作完成中断标志。

2.4.21.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgFinished (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.22 LL_FLASH_IsActiveFlag_Busy

2.4.22.1 功能介绍

检查 Flash 忙状态标志是否使能。

2.4.22.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_Busy (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.23 LL_FLASH_IsActiveFlag_ProgEraseSeqError

2.4.23.1 功能介绍

检查 Flash 编程/擦除序列错误标志是否使能。

2.4.23.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgEraseSeqError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.24 LL_FLASH_ClearFlag_ProgEraseSeqError

2.4.24.1 功能介绍

清除 Flash 编程/擦除序列错误标志。

2.4.24.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgEraseSeqError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.25 LL_FLASH_IsActiveFlag_ProgMissError

2.4.25.1 功能介绍

检查 Flash 快速编程数据丢失错误标志是否使能。

2.4.25.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgMissError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.26 LL_FLASH_ClearFlag_ProgMissError

2.4.26.1 功能介绍

清除 Flash 快速编程数据丢失错误标志。

2.4.26.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgMissError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.27 LL_FLASH_IsActiveFlag_ProgSizeError

2.4.27.1 功能介绍

检查编程位宽错误标志是否使能。

2.4.27.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgSizeError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.28 LL_FLASH_ClearFlag_ProgSizeError

2.4.28.1 功能介绍

清除编程位宽错误标志。

2.4.28.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgSizeError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.29 LL_FLASH_IsActiveFlag_ProgAlignError

2.4.29.1 功能介绍

检查编程地址未对齐错误标志是否使能。

2.4.29.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgAlignError (void)
输入	无
输出	无

返回值	{0, 1}
资源使用	
说明	

2.4.30 LL_FLASH_ClearFlag_ProgAlignError

2.4.30.1 功能介绍

清除编程地址未对齐错误标志。

2.4.30.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgAlignError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.31 LL_FLASH_IsActiveFlag_ProgError

2.4.31.1 功能介绍

检查编程错误标志是否使能。

2.4.31.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_ProgError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.32 LL_FLASH_ClearFlag_ProgError

2.4.32.1 功能介绍

清除编程错误标志。

2.4.32.2 接口定义

函数接口	void LL_FLASH_ClearFlag_ProgError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.33 LL_FLASH_IsActiveFlag_WRPError

2.4.33.1 功能介绍

检查写保护错误标志是否使能。

2.4.33.2 接口定义

函数接口	uint32_t LL_FLASH_IsActiveFlag_WRPErr (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.34 LL_FLASH_ClearFlag_WRPErr

2.4.34.1 功能介绍

清除写保护错误标志。

2.4.34.2 接口定义

函数接口	void LL_FLASH_ClearFlag_WRPErr (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.35 LL_FLASH_Lock

2.4.35.1 功能介绍

锁定 FLASH 控制寄存器访问。

2.4.35.2 接口定义

函数接口	void LL_FLASH_Lock (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.36 LL_FLASH_Unlock

2.4.36.1 功能介绍

解锁 FLASH 控制寄存器访问。

2.4.36.2 接口定义

函数接口	void LL_FLASH_Unlock (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.37 LL_FLASH_IsLocked

2.4.37.1 功能介绍

检查 FLASH 控制寄存器访问是否设置。

2.4.37.2 接口定义

函数接口	uint32_t LL_FLASH_IsLocked (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.38 LL_FLASH_OB_Lock

2.4.38.1 功能介绍

锁定 FLASH 选项字节寄存器访问。

2.4.38.2 接口定义

函数接口	void LL_FLASH_OB_Lock (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.39 LL_FLASH_OB_Unlock

2.4.39.1 功能介绍

解锁 FLASH 选项字节寄存器访问。

2.4.39.2 接口定义

函数接口	void LL_FLASH_OB_Unlock (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.40 LL_FLASH_OB_IsLocked

2.4.40.1 功能介绍

检查 FLASH 选项字节寄存器访问是否设置。

2.4.40.2 接口定义

函数接口	uint32_t LL_FLASH_OB_IsLocked (void)
输入	无
输出	无

返回值	{0, 1}
资源使用	
说明	

2.4.41 LL_FLASH_EnableSecurityProtect

2.4.41.1 功能介绍

使能用户安全保护。

2.4.41.2 接口定义

函数接口	void LL_FLASH_EnableSecurityProtect (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.42 LL_FLASH_IsEnabledSecurityProtect

2.4.42.1 功能介绍

检查用户安全保护是否使能。

2.4.42.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledSecurityProtect (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.43 LL_FLASH_OB_Launch

2.4.43.1 功能介绍

使能选项字节加载。

2.4.43.2 接口定义

函数接口	void LL_FLASH_OB_Launch (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.44 LL_FLASH_OB_Launch_IsReady

2.4.44.1 功能介绍

检查选项字节是否加载就绪。

2.4.44.2 接口定义

函数接口	uint32_t LL_FLASH_OB_Launch_IsReady (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.45 LL_FLASH_EnableIT_ReadError

2.4.45.1 功能介绍

使能 PCROP 读取错误中断。

2.4.45.2 接口定义

函数接口	void LL_FLASH_EnableIT_ReadError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.46 LL_FLASH_DisableIT_ReadError

2.4.46.1 功能介绍

禁止 PCROP 读取错误中断。

2.4.46.2 接口定义

函数接口	void LL_FLASH_DisableIT_ReadError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.47 LL_FLASH_IsEnabledIT_ReadError

2.4.47.1 功能介绍

检查 PCROP 读取错误中断是否使能。

2.4.47.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledIT_ReadError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.48 LL_FLASH_EnableIT_ProgramError

2.4.48.1 功能介绍

使能编程/擦除操作异常中断。

2.4.48.2 接口定义

函数接口	void LL_FLASH_EnableIT_ProgramError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.49 LL_FLASH_DisableIT_ProgramError

2.4.49.1 功能介绍

禁止编程/擦除操作异常中断。

2.4.49.2 接口定义

函数接口	void LL_FLASH_DisableIT_ProgramError (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.50 LL_FLASH_IsEnabledIT_ProgramError

2.4.50.1 功能介绍

检查编程/擦除操作异常中断是否使能。

2.4.50.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledIT_ProgramError (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.51 LL_FLASH_EnableIT_ProgramFinish

2.4.51.1 功能介绍

使能编程/擦除操作完成中断。

2.4.51.2 接口定义

函数接口	void LL_FLASH_EnableIT_ProgramFinish (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.4.52 LL_FLASH_DisableIT_ProgramFinish

2.4.52.1 功能介绍

禁止编程/擦除操作完成中断。

2.4.52.2 接口定义

函数接口	void LL_FLASH_DisableIT_ProgramFinish (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.53 LL_FLASH_IsEnabledIT_ProgramFinish

2.4.53.1 功能介绍

检查编程/擦除操作完成中断是否使能。

2.4.53.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledIT_ProgramFinish (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.54 LL_FLASH_EnableIT

2.4.54.1 功能介绍

使能中断。

2.4.54.2 接口定义

函数接口	void LL_FLASH_EnableIT (uint32_t FlagType)
输入	uint32_t FlagType: { LL_FLASH_IT_RDE, LL_FLASH_IT_OPE, LL_FLASH_IT_EOP }
输出	无
返回值	无
资源使用	
说明	

2.4.55 LL_FLASH_DisableIT

2.4.55.1 功能介绍

禁止使能中断。

2.4.55.2 接口定义

函数接口	void LL_FLASH_DisableIT (uint32_t FlagType)
输入	uint32_t FlagType: { LL_FLASH_IT_RDE, LL_FLASH_IT_OPE, LL_FLASH_IT_EOP }
输出	无
返回值	无
资源使用	
说明	

2.4.56 LL_FLASH_IsEnabledIT

2.4.56.1 功能介绍

检查中断是否使能。

2.4.56.2 接口定义

函数接口	uint32_t LL_FLASH_IsEnabledIT (uint32_t FlagType)
输入	uint32_t FlagType: { LL_FLASH_IT_RDE, LL_FLASH_IT_OPE, LL_FLASH_IT_EOP }
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.57 LL_FLASH_SetErasePageN

2.4.57.1 功能介绍

设置 FLASH 擦除页码。

2.4.57.2 接口定义

函数接口	void LL_FLASH_SetErasePageN (uint32_t Numbers)
输入	uint32_t Numbers: [0, 0x3F]
输出	无
返回值	无
资源使用	
说明	

2.4.58 LL_FLASH_GetErasePageN

2.4.58.1 功能介绍

获取 FLASH 擦除页码。

2.4.58.2 接口定义

函数接口	uint32_t LL_FLASH_GetErasePageN (void)
输入	无
输出	无
返回值	[0, 0x3F]

资源使用	
说明	

2.4.59 LL_FLASH_OB_StartReload

2.4.59.1 功能介绍

启动选项字节更新。

2.4.59.2 接口定义

函数接口	void LL_FLASH_OB_StartReload (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.60 LL_FLASH_OBReloading_IsOngoing

2.4.60.1 功能介绍

检查选项字节是否正在更新。

2.4.60.2 接口定义

函数接口	uint32_t LL_FLASH_OBReloading_IsOngoing (void)
输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.61 LL_FLASH_EraseStart

2.4.61.1 功能介绍

启动 FLASH 擦除。

2.4.61.2 接口定义

函数接口	void LL_FLASH_EraseStart (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.4.62 LL_FLASH_Erase_IsOngoing

2.4.62.1 功能介绍

检查 FLASH 擦除是否完成。

2.4.62.2 接口定义

函数接口	uint32_t LL_FLASH_Erase_IsOngoing (void)
------	--

输入	无
输出	无
返回值	{0, 1}
资源使用	
说明	

2.4.63 LL_FLASH_SetEraseMode

2.4.63.1 功能介绍

设置 FLASH 擦除模式。

2.4.63.2 接口定义

函数接口	void LL_FLASH_SetEraseMode (uint32_t EraseMode)
输入	uint32_t EraseMode: { LL_FLASH_MODEERASE_QUIT , LL_FLASH_MODEERASE_PAGES, LL_FLASH_MODEERASE_MASS }
输出	无
返回值	无
资源使用	
说明	

2.4.64 LL_FLASH_GetEraseMode

2.4.64.1 功能介绍

获取 FLASH 擦除模式。

2.4.64.2 接口定义

函数接口	uint32_t LL_FLASH_GetEraseMode (void)
输入	无
输出	无
返回值	{ LL_FLASH_MODEERASE_QUIT , LL_FLASH_MODEERASE_PAGES, LL_FLASH_MODEERASE_MASS }
资源使用	
说明	

2.4.65 LL_FLASH_SetProgramMode

2.4.65.1 功能介绍

设置 Flash 编程模式。

2.4.65.2 接口定义

函数接口	void LL_FLASH_SetProgramMode(uint32_t ProgramMode)
输入	uint32_t ProgramMode: { LL_FLASH_MODEPROGRAM_WORD, LL_FLASH_MODEPROGRAM_FAST, LL_FLASH_MODEPROGRAM_OUT }

输出	无
返回值	无
资源使用	
说明	

2.4.66 LL_FLASH_GetProgramMode

2.4.66.1 功能介绍

获取 Flash 编程模式。

2.4.66.2 接口定义

函数接口	uint32_t LL_FLASH_GetProgramMode (void)
输入	无
输出	无
返回值	{ LL_FLASH_MODEPROGRAM_WORD, LL_FLASH_MODEPROGRAM_FAST, LL_FLASH_MODEPROGRAM_OUT }
资源使用	
说明	

2.4.67 LL_FLASH_OB_SetBORHysteresis

2.4.67.1 功能介绍

设置 FLASH 选项字节 BOR 迟滞。

2.4.67.2 接口定义

函数接口	void LL_FLASH_OB_SetBORHysteresis (uint32_t Hysteresis)
输入	uint32_t Hysteresis: { LL_FLASH_OB_BORHYS_NONE ,LL_FLASH_OB_BORHYS_15MV , LL_FLASH_OB_BORHYS_30MV , LL_FLASH_OB_BORHYS_45MV }
输出	无
返回值	无
资源使用	
说明	

2.4.68 LL_FLASH_OB_GetBORHysteresis

2.4.68.1 功能介绍

获取 FLASH 选项字节 BOR 迟滞。

2.4.68.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBORHysteresis (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_BORHYS_NONE ,LL_FLASH_OB_BORHYS_15MV , LL_FLASH_OB_BORHYS_30MV , LL_FLASH_OB_BORHYS_45MV }
资源使用	

说明	
----	--

2.4.69 LL_FLASH_OB_SetBORLevel

2.4.69.1 功能介绍

设置 FLASH 选项字节 BOR 阈值等级。

2.4.69.2 接口定义

函数接口	void LL_FLASH_OB_SetBORLevel (uint32_t Level)
输入	uint32_t Level: { LL_FLASH_OB_BORL_0, LL_FLASH_OB_BORL_2, LL_FLASH_OB_BORL_4, LL_FLASH_OB_BORL_6, LL_FLASH_OB_BORL_8, LL_FLASH_OB_BORL_10, LL_FLASH_OB_BORL_12, LL_FLASH_OB_BORL_14}
输出	无
返回值	无
资源使用	
说明	

2.4.70 LL_FLASH_OB_GetBORLevel

2.4.70.1 功能介绍

获取 FLASH 选项字节 BOR 阈值等级。

2.4.70.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBORLevel (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_BORL_0, LL_FLASH_OB_BORL_2, LL_FLASH_OB_BORL_4, LL_FLASH_OB_BORL_6, LL_FLASH_OB_BORL_8, LL_FLASH_OB_BORL_10, LL_FLASH_OB_BORL_12, LL_FLASH_OB_BORL_14}
资源使用	
说明	

2.4.71 LL_FLASH_OB_SetBORState

2.4.71.1 功能介绍

设置 FLASH 选项字节 BOR 状态。

2.4.71.2 接口定义

函数接口	void LL_FLASH_OB_SetBORState (uint32_t State)
输入	uint32_t State: { LL_FLASH_OB_BOR_DISABLE, LL_FLASH_OB_BOR_ENABLE }
输出	无
返回值	无
资源使用	
说明	

2.4.72 LL_FLASH_OB_GetBORState

2.4.72.1 功能介绍

获取 FLASH 选项字节 BOR 状态。

2.4.72.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBORState (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_BOR_DISABLE, LL_FLASH_OB_BOR_ENABLE }
资源使用	
说明	

2.4.73 LL_FLASH_OB_SetRDPLLevel

2.4.73.1 功能介绍

设置 FLASH 选项字节 RDP 保护等级。

2.4.73.2 接口定义

函数接口	void LL_FLASH_OB_SetRDPLLevel (uint32_t Level)
输入	uint32_t Level: { LL_FLASH_OB_RDP_0, LL_FLASH_OB_RDP_1, LL_FLASH_OB_RDP_2 }
输出	无
返回值	无
资源使用	
说明	

2.4.74 LL_FLASH_OB_GetRDPLLevel

2.4.74.1 功能介绍

获取 FLASH 选项字节 RDP 保护等级。

2.4.74.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetRDPLLevel (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_RDP_0, LL_FLASH_OB_RDP_1, LL_FLASH_OB_RDP_2 }
资源使用	
说明	

2.4.75 LL_FLASH_OB_SetBOOTSEL

2.4.75.1 功能介绍

设置 FLASH 选项字节启动配置选择。

2.4.75.2 接口定义

函数接口	void LL_FLASH_OB_SetBOOTSEL (uint32_t Selection)
------	--

输入	uint32_t Selection: { LL_FLASH_OB_BOOTSEL_BOOT0PIN, LL_FLASH_OB_BOOTSEL_BOOT0SW }
输出	无
返回值	无
资源使用	
说明	

2.4.76 LL_FLASH_OB_GetBOOTSEL

2.4.76.1 功能介绍

获取 FLASH 选项字节启动配置选择。

2.4.76.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBOOTSEL (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_BOOTSEL_BOOT0PIN, LL_FLASH_OB_BOOTSEL_BOOT0SW }
资源使用	
说明	

2.4.77 LL_FLASH_OB_SetBOOT0SW

2.4.77.1 功能介绍

设置 FLASH 选项字节启动条件选择。

2.4.77.2 接口定义

函数接口	void LL_FLASH_OB_SetBOOT0SW (uint32_t Selection)
输入	uint32_t Selection: { LL_FLASH_OB_BOOT0SW_BOOTLOADER, LL_FLASH_OB_BOOT0SW_USERFLASH }
输出	无
返回值	无
资源使用	
说明	

2.4.78 LL_FLASH_OB_GetBOOT0SW

2.4.78.1 功能介绍

获取 FLASH 选项字节启动条件选择。

2.4.78.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBOOT0SW (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_BOOT0SW_BOOTLOADER, LL_FLASH_OB_BOOT0SW_USERFLASH }

资源使用	
说明	

2.4.79 LL_FLASH_OB_SetResetPin

2.4.79.1 功能介绍

设置复位管脚选择。

2.4.79.2 接口定义

函数接口	void LL_FLASH_OB_SetResetPin (uint32_t PinType)
输入	uint32_t PinType: { LL_FLASH_OB_nRST_ENABLE, LL_FLASH_OB_nRST_DISABLE }
输出	无
返回值	无
资源使用	
说明	

2.4.80 LL_FLASH_OB_GetResetPin

2.4.80.1 功能介绍

获取复位管脚选择。

2.4.80.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetResetPin (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_nRST_ENABLE, LL_FLASH_OB_nRST_DISABLE }
资源使用	
说明	

2.4.81 LL_FLASH_OB_SetBootSize

2.4.81.1 功能介绍

设置 System memory 区大小。

2.4.81.2 接口定义

函数接口	void LL_FLASH_OB_SetBootSize (uint32_t Size)
输入	uint32_t Size: { LL_FLASH_OB_BOOTSIZE_2, LL_FLASH_OB_BOOTSIZE_4, LL_FLASH_OB_BOOTSIZE_6, LL_FLASH_OB_BOOTSIZE_8, LL_FLASH_OB_BOOTSIZE_10, LL_FLASH_OB_BOOTSIZE_12, LL_FLASH_OB_BOOTSIZE_14, LL_FLASH_OB_BOOTSIZE_16 }
输出	无
返回值	无
资源使用	
说明	

2.4.82 LL_FLASH_OB_GetBootSize

2.4.82.1 功能介绍

获取 System memory 区大小。

2.4.82.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBootSize (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_BOOTSIZE_2, LL_FLASH_OB_BOOTSIZE_4, LL_FLASH_OB_BOOTSIZE_6, LL_FLASH_OB_BOOTSIZE_8, LL_FLASH_OB_BOOTSIZE_10, LL_FLASH_OB_BOOTSIZE_12, LL_FLASH_OB_BOOTSIZE_14, LL_FLASH_OB_BOOTSIZE_16 }
资源使用	
说明	

2.4.83 LL_FLASH_OB_SetIWDGTypeInStopMode

2.4.83.1 功能介绍

设置独立看门狗在 Stop 模式下计数器是否停止控制。

2.4.83.2 接口定义

函数接口	void LL_FLASH_OB_SetIWDGTypeInStopMode (uint32_t Selection)
输入	uint32_t Selection: { LL_FLASH_OB_IWDGTYPE_STOP, LL_FLASH_OB_IWDGTYPE_NORMAL }
输出	无
返回值	无
资源使用	
说明	

2.4.84 LL_FLASH_OB_GetIWDGTypeInStopMode

2.4.84.1 功能介绍

获取独立看门狗在 Stop 模式下计数器是否停止控制。

2.4.84.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetIWDGTypeInStopMode (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_IWDGTYPE_STOP, LL_FLASH_OB_IWDGTYPE_NORMAL }
资源使用	
说明	

2.4.85 LL_FLASH_OB_SetIWDGSourceType

2.4.85.1 功能介绍

设置独立看门狗的使能方式。

2.4.85.2 接口定义

函数接口	void LL_FLASH_OB_SetIWDGSourceType (uint32_t Selection)
输入	uint32_t Selection: { LL_FLASH_OB_IWDGSOURCETYPE_HARDWARE, LL_FLASH_OB_IWDGSOURCETYPE_SOFTWARE }
输出	无
返回值	无
资源使用	
说明	

2.4.86 LL_FLASH_OB_GetIWDGSourceType

2.4.86.1 功能介绍

获取独立看门狗的使能方式。

2.4.86.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetIWDGSourceType (void)
输入	无
输出	无
返回值	{ LL_FLASH_OB_IWDGSOURCETYPE_HARDWARE, LL_FLASH_OB_IWDGSOURCETYPE_SOFTWARE }
资源使用	
说明	

2.4.87 LL_FLASH_OB_SetPCROPStartPage

2.4.87.1 功能介绍

设置 PCROP1A 保护区域的起始地址页号。

2.4.87.2 接口定义

函数接口	void LL_FLASH_OB_SetPCROPStartPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.88 LL_FLASH_OB_GetPCROPStartPage

2.4.88.1 功能介绍

获取 PCROP1A 保护区域的起始地址页号。

2.4.88.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetPCROPStartPage (void)
输入	无
输出	无

返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.89 LL_FLASH_OB_SetPCROPAEndPage

2.4.89.1 功能介绍

设置 PCROP1A 保护区域的结束地址页号。

2.4.89.2 接口定义

函数接口	void LL_FLASH_OB_SetPCROPAEndPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.90 LL_FLASH_OB_GetPCROPAEndPage

2.4.90.1 功能介绍

获取 PCROP1A 保护区域的结束地址页号。

2.4.90.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetPCROPAEndPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.91 LL_FLASH_OB_SetPCROPBStartPage

2.4.91.1 功能介绍

设置 PCROP1B 保护区域的起始地址页号。

2.4.91.2 接口定义

函数接口	void LL_FLASH_OB_SetPCROPBStartPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.92 LL_FLASH_OB_GetPCROPBStartPage

2.4.92.1 功能介绍

获取 PCROP1B 保护区域的起始地址页号。

2.4.92.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetPCROPBStartPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.93 LL_FLASH_OB_SetPCROPBEndPage

2.4.93.1 功能介绍

设置 PCROP1B 保护区域的结束地址页号。

2.4.93.2 接口定义

函数接口	void LL_FLASH_OB_SetPCROPBEndPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.94 LL_FLASH_OB_GetPCROPBEndPage

2.4.94.1 功能介绍

获取 PCROP1B 保护区域的结束地址页号。

2.4.94.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetPCROPBEndPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.95 LL_FLASH_OB_SetWRPASTartPage

2.4.95.1 功能介绍

设置 WRP1A 保护区域的起始地址页号。

2.4.95.2 接口定义

函数接口	void LL_FLASH_OB_SetWRPASTartPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	

说明	
----	--

2.4.96 LL_FLASH_OB_GetWRPStartPage

2.4.96.1 功能介绍

获取 WRP1A 保护区域的起始地址页号。

2.4.96.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetWRPStartPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.97 LL_FLASH_OB_SetWRPEndPage

2.4.97.1 功能介绍

设置 WRP1A 保护区域的结束地址页号。

2.4.97.2 接口定义

函数接口	void LL_FLASH_OB_SetWRPEndPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.98 LL_FLASH_OB_GetWRPEndPage

2.4.98.1 功能介绍

获取 WRP1A 保护区域的结束地址页号。

2.4.98.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetWRPEndPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.99 LL_FLASH_OB_SetWRPStartPage

2.4.99.1 功能介绍

设置 WRP1B 保护区域的起始地址页号。

2.4.99.2 接口定义

函数接口	void LL_FLASH_OB_SetWRPStartPage (uint32_t Page)
------	--

输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.100LL_FLASH_OB_GetWRPBStartPage

2.4.100.1 功能介绍

获取 WRP1B 保护区域的起始地址页号。

2.4.100.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetWRPBStartPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.101LL_FLASH_OB_SetWRPBEndPage

2.4.101.1 功能介绍

设置 WRP1B 保护区域的结束地址页号。

2.4.101.2 接口定义

函数接口	void LL_FLASH_OB_SetWRPBEndPage (uint32_t Page)
输入	uint32_t Page: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.102LL_FLASH_OB_GetWRPBEndPage

2.4.102.1 功能介绍

获取 WRP1B 保护区域的结束地址页号。

2.4.102.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetWRPBEndPage (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.103 LL_FLASH_OB_SetSecSize

2.4.103.1 功能介绍

设置用户安全区域占用 User flash 页的数量。

2.4.103.2 接口定义

函数接口	void LL_FLASH_OB_SetSecSize (uint32_t Size)
输入	uint32_t Size: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.4.104 LL_FLASH_OB_GetSecSize

2.4.104.1 功能介绍

获取用户安全区域占用 User flash 页的数量。

2.4.104.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetSecSize (void)
输入	无
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.4.105 LL_FLASH_OB_SetBootLock

2.4.105.1 功能介绍

设置是否强制从 User flash 启动。

2.4.105.2 接口定义

函数接口	void LL_FLASH_OB_SetBootLock (uint32_t Selection)
输入	uint32_t Selection: { LL_FLASH_OB_BOOTLOCK_NONE, LL_FLASH_OB_BOOTLOCK_USERFLASH }
输出	无
返回值	无
资源使用	
说明	

2.4.106 LL_FLASH_OB_GetBootLock

2.4.106.1 功能介绍

获取是否强制从 User flash 启动。

2.4.106.2 接口定义

函数接口	uint32_t LL_FLASH_OB_GetBootLock (void)
------	---

输入	无
输出	无
返回值	{ LL_FLASH_OB_BOOTLOCK_NONE, LL_FLASH_OB_BOOTLOCK_USERFLASH }
资源使用	
说明	

2.5 GPIO 模块

属性	类型	字段名	含义
GPIO_TypeDef			
读写	uint32_t	MODER	端口模式寄存器
读写	uint32_t	OTYPER	端口输出类型寄存器
读写	uint32_t	PUPDR	端口上拉/下拉寄存器
只读	uint32_t	IDR	端口输入寄存器
读写	uint32_t	ODR	端口输出数据寄存器
只写	uint32_t	BSRR	端口置位/复位寄存器
读写	uint32_t	LCKR	端口配置锁定寄存器
只写	uint32_t	AFRL	复用功能低位寄存器
只写	uint32_t	AFRH	复用功能高位寄存器
只写	uint32_t	BRR	端口位复位寄存器
LL_GPIO_InitTypeDef			
读写	uint32_t	Pin	需要配置的 GPIO 引脚
读写	uint32_t	Mode	所选引脚的工作模式
读写	uint32_t	OutputType	所选引脚的操作输出类型
读写	uint32_t	Pull	选定引脚的操作上拉/下拉

读写	uint32_t	Alternate	外设要连接到所选引脚
----	----------	-----------	------------

2.5.1 LL_GPIO_SetPinMode

2.5.1.1 功能介绍

设置 GPIO 端口引脚的模式。

2.5.1.2 接口定义

函数接口	void LL_GPIO_SetPinMode (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Mode)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15} uint32_t Mode: { LL_GPIO_MODE_INPUT, LL_GPIO_MODE_OUTPUT, LL_GPIO_MODE_ALTERNATE, LL_GPIO_MODE_ANALOG }
输出	无
返回值	无
资源使用	
说明	

2.5.2 LL_GPIO_GetPinMode

2.5.2.1 功能介绍

获取 GPIO 端口引脚的模式。

2.5.2.2 接口定义

函数接口	uint32_t LL_GPIO_GetPinMode (GPIO_TypeDef * GPIOx, uint32_t Pin)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	{ LL_GPIO_MODE_INPUT, LL_GPIO_MODE_OUTPUT, LL_GPIO_MODE_ALTERNATE, LL_GPIO_MODE_ANALOG }
资源使用	

说明	
----	--

2.5.3 LL_GPIO_SetPinOutputType

2.5.3.1 功能介绍

设置 GPIO 端口引脚的输出类型。

2.5.3.2 接口定义

函数接口	void LL_GPIO_SetPinOutputType (GPIO_TypeDef * GPIOx, uint32_t PinMask, uint32_t OutputType)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL } uint32_t OutputType: { LL_GPIO_OUTPUT_PUSH_PULL, LL_GPIO_OUTPUT_OPENDRAIN }
输出	无
返回值	无
资源使用	
说明	

2.5.4 LL_GPIO_GetPinOutputType

2.5.4.1 功能介绍

获取 GPIO 端口引脚的输出类型。

2.5.4.2 接口定义

函数接口	uint32_t LL_GPIO_GetPinOutputType(GPIO_TypeDef * GPIOx, uint32_t Pin)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	{ LL_GPIO_OUTPUT_PUSH_PULL,

	LL_GPIO_OUTPUT_OPENDRAIN }
资源使用	
说明	

2.5.5 LL_GPIO_SetPinPull

2.5.5.1 功能介绍

设置 GPIO 端口引脚的上拉\下拉。

2.5.5.2 接口定义

函数接口	void LL_GPIO_SetPinPull (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Pull)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15} uint32_t Pull: { LL_GPIO_PULL_NO, LL_GPIO_PULL_UP, LL_GPIO_PULL_DOWN }
输出	无
返回值	无
资源使用	
说明	

2.5.6 LL_GPIO_GetPinPull

2.5.6.1 功能介绍

获取 GPIO 端口引脚的上拉\下拉。

2.5.6.2 接口定义

函数接口	uint32_t LL_GPIO_GetPinPull (GPIO_TypeDef * GPIOx, uint32_t Pin)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	{ LL_GPIO_PULL_NO, LL_GPIO_PULL_UP, LL_GPIO_PULL_DOWN }
资源使用	

说明	
----	--

2.5.7 LL_GPIO_SetAFPin_0_7

2.5.7.1 功能介绍

设置 GPIO 端口引脚 0~7 的复用。

2.5.7.2 接口定义

函数接口	void LL_GPIO_SetAFPin_0_7 (GPIO_TypeDef * GPIOx, uint32_t Pin, uint32_t Alternate)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7 } uint32_t Alternate: { LL_GPIO_AF_0, LL_GPIO_AF_1, LL_GPIO_AF_2, LL_GPIO_AF_3, LL_GPIO_AF_4, LL_GPIO_AF_5, LL_GPIO_AF_6, LL_GPIO_AF_7 }
输出	无
返回值	无
资源使用	
说明	

2.5.8 LL_GPIO_GetAFPin_0_7

2.5.8.1 功能介绍

获取 GPIO 端口引脚 0~7 的复用。

2.5.8.2 接口定义

函数接口	uint32_t LL_GPIO_GetAFPin_0_7 (GPIO_TypeDef * GPIOx, uint32_t Pin)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7 }
输出	无
返回值	{ LL_GPIO_AF_0, LL_GPIO_AF_1, LL_GPIO_AF_2, LL_GPIO_AF_3, LL_GPIO_AF_4, LL_GPIO_AF_5, LL_GPIO_AF_6, LL_GPIO_AF_7 }
资源使用	
说明	

2.5.9 LL_GPIO_SetAFPin_8_15

2.5.9.1 功能介绍

设置 GPIO 端口引脚 8~15 的复用。

2.5.9.2 接口定义

函数接口	void LL_GPIO_SetAFPin_8_15 (GPIO_TypeDef * GPIOx, uint32_t
------	---

	Pin, uint32_t Alternate)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15} uint32_t Alternate: { LL_GPIO_AF_0, LL_GPIO_AF_1, LL_GPIO_AF_2, LL_GPIO_AF_3, LL_GPIO_AF_4, LL_GPIO_AF_5, LL_GPIO_AF_6, LL_GPIO_AF_7 }
输出	无
返回值	无
资源使用	
说明	

2.5.10 LL_GPIO_GetAFPin_8_15

2.5.10.1 功能介绍

获取 GPIO 端口引脚 8~15 的复用。

2.5.10.2 接口定义

函数接口	uint32_t LL_GPIO_GetAFPin_8_15 (GPIO_TypeDef * GPIOx, uint32_t Pin)
输入	GPIO_TypeDef * GPIOx uint32_t Pin: { LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	{ LL_GPIO_AF_0, LL_GPIO_AF_1, LL_GPIO_AF_2, LL_GPIO_AF_3, LL_GPIO_AF_4, LL_GPIO_AF_5, LL_GPIO_AF_6, LL_GPIO_AF_7 }
资源使用	
说明	

2.5.11 LL_GPIO_LockPin

2.5.11.1 功能介绍

锁定 GPIO 端口的引脚的配置。

2.5.11.2 接口定义

函数接口	void LL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14,

	LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	无
资源使用	
说明	

2.5.12 LL_GPIO_IsPinLocked

2.5.12.1 功能介绍

检查指定 GPIO 端口的引脚的是否被锁定。

2.5.12.2 接口定义

函数接口	uint32_t LL_GPIO_IsPinLocked (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	{0, 1}
资源使用	
说明	

2.5.13 LL_GPIO_IsAnyPinLocked

2.5.13.1 功能介绍

检查 GPIO 端口的所有引脚的是否被锁定。

2.5.13.2 接口定义

函数接口	uint32_t LL_GPIO_IsAnyPinLocked (GPIO_TypeDef * GPIOx)
输入	GPIO_TypeDef * GPIOx
输出	无
返回值	{0, 1}
资源使用	
说明	

2.5.14 LL_GPIO_ReadInputPort

2.5.14.1 功能介绍

读取 GPIO 端口的输入数据寄存器值。

2.5.14.2 接口定义

函数接口	uint32_t LL_GPIO_ReadInputPort (GPIO_TypeDef * GPIOx)
输入	GPIO_TypeDef * GPIOx

输出	无
返回值	输入数据
资源使用	
说明	

2.5.15 LL_GPIO_IsInputPinSet

2.5.15.1 功能介绍

检查 GPIO 端口输入引脚是否为高电平。

2.5.15.2 接口定义

函数接口	uint32_t LL_GPIO_IsInputPinSet (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	{0, 1}
资源使用	
说明	

2.5.16 LL_GPIO_WriteOutputPort

2.5.16.1 功能介绍

写 GPIO 端口输出寄存器的值。

2.5.16.2 接口定义

函数接口	void LL_GPIO_WriteOutputPort (GPIO_TypeDef * GPIOx, uint32_t PortValue)
输入	GPIO_TypeDef * GPIOx uint32_t PortValue
输出	无
返回值	无
资源使用	
说明	

2.5.17 LL_GPIO_ReadOutputPort

2.5.17.1 功能介绍

读取 GPIO 端口输出数据寄存器值。

2.5.17.2 接口定义

函数接口	uint32_t LL_GPIO_ReadOutputPort (GPIO_TypeDef * GPIOx)
------	--

输入	GPIO_TypeDef * GPIOx
输出	无
返回值	输出数据寄存器值
资源使用	
说明	

2.5.18 LL_GPIO_IsOutputPinSet

2.5.18.1 功能介绍

检查 GPIO 输出端口引脚是否为高电平。

2.5.18.2 接口定义

函数接口	uint32_t LL_GPIO_IsOutputPinSet (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	{0, 1}
资源使用	
说明	

2.5.19 LL_GPIO_SetOutputPin

2.5.19.1 功能介绍

GPIO 输出端口引脚设置为高电平。

2.5.19.2 接口定义

函数接口	void LL_GPIO_SetOutputPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	无
资源使用	
说明	

2.5.20 LL_GPIO_ResetOutputPin

2.5.20.1 功能介绍

GPIO 输出端口引脚设置为低电平。

2.5.20.2 接口定义

函数接口	void LL_GPIO_ResetOutputPin (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	无
资源使用	
说明	

2.5.21 LL_GPIO_TogglePin

2.5.21.1 功能介绍

翻转 GPIO 端口引脚电平。

2.5.21.2 接口定义

函数接口	void LL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint32_t PinMask)
输入	GPIO_TypeDef * GPIOx uint32_t PinMask: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15, LL_GPIO_PIN_ALL }
输出	无
返回值	无
资源使用	
说明	

2.5.22 LL_GPIO_DeInit

2.5.22.1 功能介绍

清除 GPIO 初始化配置。

2.5.22.2 接口定义

函数接口	ErrorStatus LL_GPIO_DeInit (GPIO_TypeDef * GPIOx)
输入	GPIO_TypeDef * GPIOx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.5.23 LL_GPIO_Init

2.5.23.1 功能介绍

GPIO 初始化。

2.5.23.2 接口定义

函数接口	ErrorStatus LL_GPIO_Init (GPIO_TypeDef * GPIOx, LL_GPIO_InitTypeDef * GPIO_InitStruct)
输入	GPIO_TypeDef * GPIOx LL_GPIO_InitTypeDef * GPIO_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.5.24 LL_GPIO_StructInit

2.5.24.1 功能介绍

GPIO 结构体初始化。

2.5.24.2 接口定义

函数接口	void LL_GPIO_StructInit (LL_GPIO_InitTypeDef * GPIO_InitStruct)
输入	LL_GPIO_InitTypeDef * GPIO_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.6 PWR 模块

2.6.1 LL_PWR_EnableBGR

2.6.1.1 功能介绍

使能 BGR。

2.6.1.2 接口定义

函数接口	void LL_PWR_EnableBGR (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.6.2 LL_PWR_DisableBGR

2.6.2.1 功能介绍

禁止使能 BGR。

2.6.2.2 接口定义

函数接口	void LL_PWR_DisableBGR (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.3 LL_PWR_IsEnabledBGR

2.6.3.1 功能介绍

检查是否使能 BGR。

2.6.3.2 接口定义

函数接口	uint32_t LL_PWR_IsEnabledBGR (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.4 LL_PWR_SetVref

2.6.4.1 功能介绍

设置电压参考选择。

2.6.4.2 接口定义

函数接口	void LL_PWR_SetVref (uint32_t Vref)
输入	uint32_t Vref: { LL_PWR_VREF_1V25, LL_PWR_VREF_2V5, LL_PWR_VREF_3V, LL_PWR_VREF_4V }
输出	无
返回值	无
资源使用	
说明	

2.6.5 LL_PWR_GetVref

2.6.5.1 功能介绍

设置电压参考选择。

2.6.5.2 接口定义

函数接口	uint32_t LL_PWR_GetVref (void)
输入	无
输出	无
返回值	{ LL_PWR_VREF_1V25, LL_PWR_VREF_2V5, LL_PWR_VREF_3V, LL_PWR_VREF_4V }
资源使用	
说明	

2.6.6 LL_PWR_EnableBORPORIntermittentDetection

2.6.6.1 功能介绍

使能 BOR POR 间歇检测。

2.6.6.2 接口定义

函数接口	void LL_PWR_EnableBORPORIntermittentDetection (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.7 LL_PWR_DisableBORPORIntermittentDetection

2.6.7.1 功能介绍

禁止使能 BOR POR 间歇检测。

2.6.7.2 接口定义

函数接口	void LL_PWR_DisableBORPORIntermittentDetection (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.8 LL_PWR_IsEnabledBORPORIntermittentDetection

2.6.8.1 功能介绍

检查是否使能 BOR POR 间歇检测。

2.6.8.2 接口定义

函数接口	uint32_t LL_PWR_IsEnabledBORPORIntermittentDetection (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.9 LL_PWR_EnableLowPowerRunMode

2.6.9.1 功能介绍

使能低功耗模式。

2.6.9.2 接口定义

函数接口	void LL_PWR_EnableLowPowerRunMode (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.10 LL_PWR_DisableLowPowerRunMode

2.6.10.1 功能介绍

禁止使能低功耗模式。

2.6.10.2 接口定义

函数接口	void LL_PWR_DisableLowPowerRunMode (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.11 LL_PWR_IsEnabledLowPowerRunMode

2.6.11.1 功能介绍

检查是否使能低功耗模式。

2.6.11.2 接口定义

函数接口	uint32_t LL_PWR_IsEnabledLowPowerRunMode (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.12 LL_PWR_SetPowerMode

2.6.12.1 功能介绍

设置低功耗模式。

2.6.12.2 接口定义

函数接口	void LL_PWR_SetPowerMode (uint32_t LowPowerMode)
输入	uint32_t LowPowerMode: { LL_PWR_MODE_STOP, LL_PWR_MODE_NOSTOP }

输出	无
返回值	无
资源使用	
说明	

2.6.13 LL_PWR_GetPowerMode

2.6.13.1 功能介绍

获取低功耗模式。

2.6.13.2 接口定义

函数接口	uint32_t LL_PWR_GetPowerMode (void)
输入	无
输出	无
返回值	{ LL_PWR_MODE_STOP, LL_PWR_MODE_NOSTOP }
资源使用	
说明	

2.6.14 LL_PWR_PVD_SetLevel

2.6.14.1 功能介绍

设置 PVD 阈值等级。

2.6.14.2 接口定义

函数接口	void LL_PWR_PVD_SetLevel (uint32_t PVDLevel)
输入	uint32_t PVDLevel: { LL_PWR_PVDLEVEL_0, LL_PWR_PVDLEVEL_1, LL_PWR_PVDLEVEL_2, LL_PWR_PVDLEVEL_3, LL_PWR_PVDLEVEL_4, LL_PWR_PVDLEVEL_5, LL_PWR_PVDLEVEL_6, LL_PWR_PVDLEVEL_7, LL_PWR_PVDLEVEL_8, LL_PWR_PVDLEVEL_9, LL_PWR_PVDLEVEL_10, LL_PWR_PVDLEVEL_11, LL_PWR_PVDLEVEL_12, LL_PWR_PVDLEVEL_13, LL_PWR_PVDLEVEL_14, LL_PWR_PVDLEVEL_15, LL_PWR_PVDLEVEL_16, LL_PWR_PVDLEVEL_17, LL_PWR_PVDLEVEL_18, LL_PWR_PVDLEVEL_19, LL_PWR_PVDLEVEL_20, LL_PWR_PVDLEVEL_21, LL_PWR_PVDLEVEL_22, LL_PWR_PVDLEVEL_23, LL_PWR_PVDLEVEL_24, LL_PWR_PVDLEVEL_25, LL_PWR_PVDLEVEL_26, LL_PWR_PVDLEVEL_27, LL_PWR_PVDLEVEL_28, LL_PWR_PVDLEVEL_29, LL_PWR_PVDLEVEL_30, LL_PWR_PVDLEVEL_31 }
输出	无
返回值	无
资源使用	
说明	

2.6.15 LL_PWR_PVD_GetLevel

2.6.15.1 功能介绍

获取 PVD 阈值等级。

2.6.15.2 接口定义

函数接口	uint32_t LL_PWR_PVD_GetLevel (void)
输入	无
输出	无
返回值	{ LL_PWR_PVDLEVEL_0, LL_PWR_PVDLEVEL_1, LL_PWR_PVDLEVEL_2, LL_PWR_PVDLEVEL_3, LL_PWR_PVDLEVEL_4, LL_PWR_PVDLEVEL_5, LL_PWR_PVDLEVEL_6, LL_PWR_PVDLEVEL_7, LL_PWR_PVDLEVEL_8, LL_PWR_PVDLEVEL_9, LL_PWR_PVDLEVEL_10, LL_PWR_PVDLEVEL_11, LL_PWR_PVDLEVEL_12, LL_PWR_PVDLEVEL_13, LL_PWR_PVDLEVEL_14, LL_PWR_PVDLEVEL_15, LL_PWR_PVDLEVEL_16, LL_PWR_PVDLEVEL_17, LL_PWR_PVDLEVEL_18, LL_PWR_PVDLEVEL_19, LL_PWR_PVDLEVEL_20, LL_PWR_PVDLEVEL_21, LL_PWR_PVDLEVEL_22, LL_PWR_PVDLEVEL_23, LL_PWR_PVDLEVEL_24, LL_PWR_PVDLEVEL_25, LL_PWR_PVDLEVEL_26, LL_PWR_PVDLEVEL_27, LL_PWR_PVDLEVEL_28, LL_PWR_PVDLEVEL_29, LL_PWR_PVDLEVEL_30, LL_PWR_PVDLEVEL_31 }
资源使用	
说明	

2.6.16 LL_PWR_PVD_SetSelection

2.6.16.1 功能介绍

设置 PVD 监测信号源选择。

2.6.16.2 接口定义

函数接口	void LL_PWR_PVD_SetSelection (uint32_t PVDSelction)
输入	uint32_t PVDSelction: { LL_PWR_PVDSEL_PD1, LL_PWR_PVDSEL_PD4, LL_PWR_PVDSEL_PD6, LL_PWR_PVDSEL_VDD }
输出	无
返回值	无
资源使用	
说明	

2.6.17 LL_PWR_PVD_GetSelection

2.6.17.1 功能介绍

获取 PVD 监测信号源选择。

2.6.17.2 接口定义

函数接口	uint32_t LL_PWR_PVD_GetSelection (void)
输入	无
输出	无
返回值	{ LL_PWR_PVDSEL_PD1, LL_PWR_PVDSEL_PD4, LL_PWR_PVDSEL_PD6, LL_PWR_PVDSEL_VDD }
资源使用	
说明	

2.6.18 LL_PWR_PVD_SetHysteresis

2.6.18.1 功能介绍

设置 PVD 迟滞。

2.6.18.2 接口定义

函数接口	void LL_PWR_PVD_SetHysteresis (uint32_t Hysteresis)
输入	uint32_t Hysteresis: { LL_PWR_PVDHYS_NONE, LL_PWR_PVDHYS_15MV, LL_PWR_PVDHYS_30MV, LL_PWR_PVDHYS_45MV }
输出	无
返回值	无
资源使用	
说明	

2.6.19 LL_PWR_PVD_GetHysteresis

2.6.19.1 功能介绍

获取 PVD 迟滞。

2.6.19.2 接口定义

函数接口	uint32_t LL_PWR_PVD_GetHysteresis (void)
输入	无
输出	无
返回值	{ LL_PWR_PVDHYS_NONE, LL_PWR_PVDHYS_15MV, LL_PWR_PVDHYS_30MV, LL_PWR_PVDHYS_45MV }
资源使用	
说明	

2.6.20 LL_PWR_PVD_SetFilterTime

2.6.20.1 功能介绍

设置 PVD 监测信号数字滤波时间。

2.6.20.2 接口定义

函数接口	void LL_PWR_PVD_SetFilterTime (uint32_t PVDFilterTime)
输入	uint32_t PVDFilterTime: { LL_PWR_PVDFILTERTIME_2, LL_PWR_PVDFILTERTIME_4, LL_PWR_PVDFILTERTIME_8, LL_PWR_PVDFILTERTIME_16,

	LL_PWR_PVDFILTERTIME_32, LL_PWR_PVDFILTERTIME_64, LL_PWR_PVDFILTERTIME_128, LL_PWR_PVDFILTERTIME_256 }
输出	无
返回值	无
资源使用	
说明	

2.6.21 LL_PWR_PVD_GetFilterTime

2.6.21.1 功能介绍

获取 PVD 监测信号数字滤波时间。

2.6.21.2 接口定义

函数接口	uint32_t LL_PWR_PVD_GetFilterTime (void)
输入	无
输出	无
返回值	{ LL_PWR_PVDFILTERTIME_2, LL_PWR_PVDFILTERTIME_4, LL_PWR_PVDFILTERTIME_8, LL_PWR_PVDFILTERTIME_16, LL_PWR_PVDFILTERTIME_32, LL_PWR_PVDFILTERTIME_64, LL_PWR_PVDFILTERTIME_128, LL_PWR_PVDFILTERTIME_256 }
资源使用	
说明	

2.6.22 LL_PWR_PVD_Enable

2.6.22.1 功能介绍

使能 PVD。

2.6.22.2 接口定义

函数接口	void LL_PWR_PVD_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.23 LL_PWR_PVD_Disable

2.6.23.1 功能介绍

禁止使能 PVD。

2.6.23.2 接口定义

函数接口	void LL_PWR_PVD_Disable (void)
输入	无
输出	无
返回值	无
资源使用	

说明	
----	--

2.6.24 LL_PWR_PVD_IsEnabled

2.6.24.1 功能介绍

检查是否使能 PVD。

2.6.24.2 接口定义

函数接口	uint32_t LL_PWR_PVD_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.25 LL_PWR_PVD_EnableIT

2.6.25.1 功能介绍

使能 PVD 中断。

2.6.25.2 接口定义

函数接口	void LL_PWR_PVD_EnableIT (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.26 LL_PWR_PVD_DisableIT

2.6.26.1 功能介绍

禁止使能 PVD 中断。

2.6.26.2 接口定义

函数接口	void LL_PWR_PVD_DisableIT (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.27 LL_PWR_PVD_IsEnabledIT

2.6.27.1 功能介绍

检查是否使能 PVD 中断。

2.6.27.2 接口定义

函数接口	uint32_t LL_PWR_PVD_IsEnabledIT (void)
输入	无

输出	无
返回值	{0,1}
资源使用	
说明	

2.6.28 LL_PWR_PVD_EnableThreshold

2.6.28.1 功能介绍

使能 PVD 阈值模式。

2.6.28.2 接口定义

函数接口	void LL_PWR_PVD_EnableThreshold (uint32_t ThresholdMode)
输入	uint32_t ThresholdMode: { LL_PWR_PVDMODE_HIGHTHRESHOLD, LL_PWR_PVDMODE_LOWTHRESHOLD, LL_PWR_PVDMODE_RAISINGTHRESHOLD, LL_PWR_PVDMODE_FALLINGTHRESHOLD }
输出	无
返回值	无
资源使用	
说明	

2.6.29 LL_PWR_PVD_DisableThreshold

2.6.29.1 功能介绍

禁止使能 PVD 阈值模式。

2.6.29.2 接口定义

函数接口	void LL_PWR_PVD_DisableThreshold (uint32_t ThresholdMode)
输入	uint32_t ThresholdMode: { LL_PWR_PVDMODE_HIGHTHRESHOLD, LL_PWR_PVDMODE_LOWTHRESHOLD, LL_PWR_PVDMODE_RAISINGTHRESHOLD, LL_PWR_PVDMODE_FALLINGTHRESHOLD }
输出	无
返回值	无
资源使用	
说明	

2.6.30 LL_PWR_PVD_IsEnabledThreshold

2.6.30.1 功能介绍

检查是否使能 PVD 阈值模式。

2.6.30.2 接口定义

函数接口	uint32_t LL_PWR_PVD_IsEnabledThreshold (uint32_t ThresholdMode)
输入	uint32_t ThresholdMode: { LL_PWR_PVDMODE_HIGHTHRESHOLD,

	LL_PWR_PVDMODE_LOWTHRESHOLD, LL_PWR_PVDMODE_RAISINGTHRESHOLD, LL_PWR_PVDMODE_FALLINGTHRESHOLD }
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.31 LL_PWR_PVD_EnableFilter

2.6.31.1 功能介绍

使能 PVD 监测信号数字滤波。

2.6.31.2 接口定义

函数接口	void LL_PWR_PVD_EnableFilter (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.32 LL_PWR_PVD_DisableFilter

2.6.32.1 功能介绍

禁止使能 PVD 监测信号数字滤波。

2.6.32.2 接口定义

函数接口	void LL_PWR_PVD_DisableFilter (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.33 LL_PWR_PVD_IsEnabledFilter

2.6.33.1 功能介绍

检查是否使能 PVD 监测信号数字滤波。

2.6.33.2 接口定义

函数接口	uint32_t LL_PWR_PVD_IsEnabledFilter (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.34 LL_PWR_PVD_IsActiveFlagIT

2.6.34.1 功能介绍

检查 PVD 报警中断状态标志是否置位。

2.6.34.2 接口定义

函数接口	uint32_t LL_PWR_PVD_IsActiveFlagIT (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.6.35 LL_PWR_PVD_ClearFlagIT

2.6.35.1 功能介绍

清除 PVD 报警中断状态标志。

2.6.35.2 接口定义

函数接口	void LL_PWR_PVD_ClearFlagIT (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.6.36 LL_PWR_PVD_GetStatus

2.6.36.1 功能介绍

获取 PVD 状态。

2.6.36.2 接口定义

函数接口	uint32_t LL_PWR_PVD_GetStatus (void)
输入	无
输出	无
返回值	{ LL_PWR_PVDSTATUS_HIGH, LL_PWR_PVDSTATUS_LOW }
资源使用	
说明	

2.6.37 LL_PWR_DeInit

2.6.37.1 功能介绍

清除初始化参数配置。

2.6.37.2 接口定义

函数接口	ErrorStatus LL_PWR_DeInit (void)
输入	无
输出	无

返回值	ErrorStatus
资源使用	
说明	

2.7 RTC 模块

属性	类型	字段名	含义
RTC_TypeDef			
读写	uint32_t	CR0	控制寄存器 0
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	SEC	秒计数寄存器
读写	uint32_t	MIN	分计数寄存器
读写	uint32_t	HOUR	时计数寄存器
读写	uint32_t	DAY	日计数寄存器
读写	uint32_t	WEEK	周计数寄存器
读写	uint32_t	MON	月计数寄存器
读写	uint32_t	YEAR	年计数寄存器
读写	uint32_t	ALMMIN	分闹钟寄存器
读写	uint32_t	ALMHOUR	时闹钟寄存器
读写	uint32_t	ALM WEEK	周闹钟寄存器
读写	uint32_t	COMPEN	时钟校准寄存器
LL_RTC_InitTypeDef			
读写	uint32_t	HourFormat	RTC 小时格式
读写	uint32_t	PeriodSource	RTC 周期中断源
读写	uint32_t	PITS	RTC 中断周期选择
读写	uint32_t	PITX	设置产生周期中断的时间间隔，可设定的范围为 0.5 秒到 32 秒，步进为 0.5 秒
LL_RTC_TimeTypeDef			
读写	uint32_t	TimeFormat	时间格式
读写	uint8_t	Hours	时
读写	uint8_t	Minutes	分
读写	uint8_t	Seconds	秒
LL_RTC_DateTypeDef			

读写	uint8_t	WeekDay	星期
读写	uint8_t	Month	月份
读写	uint8_t	Day	日
读写	uint8_t	Year	年
LL_RTC_AlarmTypeDef			
读写	uint32_t	TimeFormat	时间格式
读写	uint8_t	AlarmWeek	星期闹钟
读写	uint8_t	AlarmHour	时闹钟
读写	uint8_t	AlarmMinute	分闹钟

2.7.1 LL_RTC_SetHourFormat

2.7.1.1 功能介绍

设置时的格式。

2.7.1.2 接口定义

函数接口	void LL_RTC_SetHourFormat (RTC_TypeDef * RTCx, uint32_t HourFormat)
输入	RTC_TypeDef * RTCx uint32_t HourFormat: {LL_RTC_HOURFORMAT_24HOUR, LL_RTC_HOURFORMAT_AMPM}
输出	无
返回值	无
资源使用	
说明	

2.7.2 LL_RTC_GetHourFormat

2.7.2.1 功能介绍

获取时的格式。

2.7.2.2 接口定义

函数接口	uint32_t LL_RTC_GetHourFormat (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{LL_RTC_HOURFORMAT_24HOUR, LL_RTC_HOURFORMAT_AMPM}
资源使用	
说明	

2.7.3 LL_RTC_SetPeriodSource

2.7.3.1 功能介绍

设置 RTC 周期中断源。

2.7.3.2 接口定义

函数接口	void LL_RTC_SetPeriodSource (RTC_TypeDef * RTCx, uint32_t Source)
输入	RTC_TypeDef * RTCx uint32_t Source: { LL_RTC_PERIODSOURCE_PRDS, LL_RTC_PERIODSOURCE_PRDX }
输出	无
返回值	无
资源使用	
说明	

2.7.4 LL_RTC_GetPeriodSource

2.7.4.1 功能介绍

获取 RTC 周期中断源。

2.7.4.2 接口定义

函数接口	uint32_t LL_RTC_GetPeriodSource (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_PERIODSOURCE_PRDS, LL_RTC_PERIODSOURCE_PRDX }
资源使用	
说明	

2.7.5 LL_RTC_SetPRDSTime

2.7.5.1 功能介绍

设置 RTC 中断率。

2.7.5.2 接口定义

函数接口	void LL_RTC_SetPRDSTime (RTC_TypeDef * RTCx, uint32_t Time)
输入	RTC_TypeDef * RTCx uint32_t Time: { LL_RTC_PRDS_NONE, LL_RTC_PRDS_0_5S, LL_RTC_PRDS_1SEC, LL_RTC_PRDS_1MIN, LL_RTC_PRDS_1HOUR, LL_RTC_PRDS_1DAY, LL_RTC_PRDS_1MON }
输出	无
返回值	无
资源使用	
说明	

2.7.6 LL_RTC_GetPRDSTime

2.7.6.1 功能介绍

获取 RTC 中断率。

2.7.6.2 接口定义

函数接口	uint32_t LL_RTC_GetPRDSTime (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_PRDS_NONE, LL_RTC_PRDS_0_5S, LL_RTC_PRDS_1SEC, LL_RTC_PRDS_1MIN, LL_RTC_PRDS_1HOUR, LL_RTC_PRDS_1DAY, LL_RTC_PRDS_1MON }
资源使用	
说明	

2.7.7 LL_RTC_SetPRDXTime

2.7.7.1 功能介绍

设置产生周期中断的时间间隔。

2.7.7.2 接口定义

函数接口	void LL_RTC_SetPRDXTime (RTC_TypeDef * RTCx, uint32_t Time)
输入	RTC_TypeDef * RTCx uint32_t Time: [0x00, 0x3F]
输出	无
返回值	无
资源使用	
说明	

2.7.8 LL_RTC_GetPRDXTime

2.7.8.1 功能介绍

获取产生周期中断的时间间隔。

2.7.8.2 接口定义

函数接口	uint32_t LL_RTC_GetPRDXTime (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x00, 0x3F]
资源使用	
说明	

2.7.9 LL_RTC_Set1HZOutputPrecision

2.7.9.1 功能介绍

设置 1Hz 输出精度。

2.7.9.2 接口定义

函数接口	void LL_RTC_Set1HZOutputPrecision (RTC_TypeDef * RTCx, uint32_t Precision)
输入	RTC_TypeDef * RTCx

	uint32_t Precision: { LL_RTC_1HZPRECISION_NORMAL, LL_RTC_1HZPRECISION_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.7.10 LL_RTC_Get1HZOutputPrecision

2.7.10.1 功能介绍

获取 1Hz 输出精度。

2.7.10.2 接口定义

函数接口	uint32_t LL_RTC_Get1HZOutputPrecision (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_1HZPRECISION_NORMAL, LL_RTC_1HZPRECISION_HIGH }
资源使用	
说明	

2.7.11 LL_RTC_EnableReadWriteMode

2.7.11.1 功能介绍

使能写入/读出模式。

2.7.11.2 接口定义

函数接口	void LL_RTC_EnableReadWriteMode (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.12 LL_RTC_DisableReadWriteMode

2.7.12.1 功能介绍

禁止使能写入/读出模式。

2.7.12.2 接口定义

函数接口	void LL_RTC_DisableReadWriteMode (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.13 LL_RTC_IsEnabledReadWriteMode

2.7.13.1 功能介绍

检查是否使能写入/读出模式。

2.7.13.2 接口定义

函数接口	uint32_t LL_RTC_IsEnabledReadWriteMode (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.14 LL_RTC_Enable1HzOutput

2.7.14.1 功能介绍

使能 1Hz 输出。

2.7.14.2 接口定义

函数接口	void LL_RTC_Enable1HzOutput (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.15 LL_RTC_Disable1HzOutput

2.7.15.1 功能介绍

禁止使能 1Hz 输出。

2.7.15.2 接口定义

函数接口	void LL_RTC_Disable1HzOutput (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.16 LL_RTC_IsEnabled1HzOutput

2.7.16.1 功能介绍

检查是否使能 1Hz 输出。

2.7.16.2 接口定义

函数接口	uint32_t LL_RTC_IsEnabled1HzOutput (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无

返回值	{0,1}
资源使用	
说明	

2.7.17 LL_RTC_Enable

2.7.17.1 功能介绍

使能 RTC。

2.7.17.2 接口定义

函数接口	void LL_RTC_Enable (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.18 LL_RTC_Disable

2.7.18.1 功能介绍

禁止使能 RTC。

2.7.18.2 接口定义

函数接口	void LL_RTC_Disable (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.19 LL_RTC_IsEnabled

2.7.19.1 功能介绍

检查是否使能 RTC。

2.7.19.2 接口定义

函数接口	uint32_t LL_RTC_IsEnabled (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.20 LL_RTC_TIME_SetHour

2.7.20.1 功能介绍

设置 BCD 格式的时钟。

2.7.20.2 接口定义

函数接口	void LL_RTC_TIME_SetHour (RTC_TypeDef * RTCx, uint32_t Hours)
输入	RTC_TypeDef * RTCx uint32_t Hours: { LL_RTC_TIME_FORMAT_AM_OR_24, LL_RTC_TIME_FORMAT_PM }
输出	无
返回值	无
资源使用	
说明	

2.7.21 LL_RTC_TIME_GetHour

2.7.21.1 功能介绍

获取 BCD 格式的时钟。

2.7.21.2 接口定义

函数接口	uint32_t LL_RTC_TIME_GetHour (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_TIME_FORMAT_AM_OR_24, LL_RTC_TIME_FORMAT_PM }
资源使用	
说明	

2.7.22 LL_RTC_TIME_SetMinute

2.7.22.1 功能介绍

设置 BCD 格式的分钟。

2.7.22.2 接口定义

函数接口	void LL_RTC_TIME_SetMinute (RTC_TypeDef * RTCx, uint32_t Minutes)
输入	RTC_TypeDef * RTCx uint32_t Minutes: [0x00, 0x59]
输出	无
返回值	无
资源使用	
说明	

2.7.23 LL_RTC_TIME_GetMinute

2.7.23.1 功能介绍

获取 BCD 格式的分钟。

2.7.23.2 接口定义

函数接口	uint32_t LL_RTC_TIME_GetMinute (RTC_TypeDef * RTCx)
------	---

输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x00, 0x59]
资源使用	
说明	

2.7.24 LL_RTC_TIME_SetSecond

2.7.24.1 功能介绍

设置 BCD 格式的秒钟。

2.7.24.2 接口定义

函数接口	void LL_RTC_TIME_SetSecond (RTC_TypeDef * RTCx, uint32_t Seconds)
输入	RTC_TypeDef * RTCx uint32_t Seconds: [0x00, 0x59]
输出	无
返回值	无
资源使用	
说明	

2.7.25 LL_RTC_TIME_GetSecond

2.7.25.1 功能介绍

获取 BCD 格式的秒钟。

2.7.25.2 接口定义

函数接口	uint32_t LL_RTC_TIME_GetSecond (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x00, 0x59]
资源使用	
说明	

2.7.26 LL_RTC_TIME_Config

2.7.26.1 功能介绍

设置 BCD 格式的时分秒。

2.7.26.2 接口定义

函数接口	void LL_RTC_TIME_Config (RTC_TypeDef * RTCx, uint32_t Hours, uint32_t Minutes, uint32_t Seconds)
输入	RTC_TypeDef * RTCx uint32_t Hours: { LL_RTC_TIME_FORMAT_AM_OR_24, LL_RTC_TIME_FORMAT_PM } uint32_t Minutes:

	[0x00, 0x59] uint32_t Seconds: [0x00, 0x59]
输出	无
返回值	无
资源使用	
说明	

2.7.27 LL_RTC_TIME_Get

2.7.27.1 功能介绍

获取 BCD 格式的时分秒。

2.7.27.2 接口定义

函数接口	uint32_t LL_RTC_TIME_Get (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	Format: 0x00HHMMSS
资源使用	
说明	

2.7.28 LL_RTC_DATE_SetYear

2.7.28.1 功能介绍

设置年份。

2.7.28.2 接口定义

函数接口	void LL_RTC_DATE_SetYear (RTC_TypeDef * RTCx, uint32_t Year)
输入	RTC_TypeDef * RTCx uint32_t Year: [0x00, 0x99]
输出	无
返回值	无
资源使用	
说明	

2.7.29 LL_RTC_DATE_GetYear

2.7.29.1 功能介绍

获取年份。

2.7.29.2 接口定义

函数接口	uint32_t LL_RTC_DATE_GetYear (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x00, 0x99]
资源使用	

说明	
----	--

2.7.30 LL_RTC_DATE_SetWeekDay

2.7.30.1 功能介绍

设置星期。

2.7.30.2 接口定义

函数接口	void LL_RTC_DATE_SetWeekDay (RTC_TypeDef * RTCx, uint32_t WeekDay)
输入	RTC_TypeDef * RTCx uint32_t WeekDay: { LL_RTC_WEEKDAY_MONDAY, LL_RTC_WEEKDAY_TUESDAY, LL_RTC_WEEKDAY_WEDNESDAY, LL_RTC_WEEKDAY_THURSDAY, LL_RTC_WEEKDAY_FRIDAY, LL_RTC_WEEKDAY_SATURDAY, LL_RTC_WEEKDAY_SUNDAY }
输出	无
返回值	无
资源使用	
说明	

2.7.31 LL_RTC_DATE_GetWeekDay

2.7.31.1 功能介绍

获取星期。

2.7.31.2 接口定义

函数接口	uint32_t LL_RTC_DATE_GetWeekDay (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_WEEKDAY_MONDAY, LL_RTC_WEEKDAY_TUESDAY, LL_RTC_WEEKDAY_WEDNESDAY, LL_RTC_WEEKDAY_THURSDAY, LL_RTC_WEEKDAY_FRIDAY, LL_RTC_WEEKDAY_SATURDAY, LL_RTC_WEEKDAY_SUNDAY }
资源使用	
说明	

2.7.32 LL_RTC_DATE_SetMonth

2.7.32.1 功能介绍

设置月份。

2.7.32.2 接口定义

函数接口	void LL_RTC_DATE_SetMonth (RTC_TypeDef * RTCx, uint32_t Month)
输入	RTC_TypeDef * RTCx uint32_t Month: { LL_RTC_MONTH_JANUARY, LL_RTC_MONTH_FEBRUARY, LL_RTC_MONTH_MARCH, LL_RTC_MONTH_APRIL,

	LL_RTC_MONTH_MAY, LL_RTC_MONTH_JUNE, LL_RTC_MONTH_JULY, LL_RTC_MONTH_AUGUST, LL_RTC_MONTH_SEPTEMBER, LL_RTC_MONTH_OCTOBER, LL_RTC_MONTH_NOVEMBER, LL_RTC_MONTH_DECEMBER }
输出	无
返回值	无
资源使用	
说明	

2.7.33 LL_RTC_DATE_GetMonth

2.7.33.1 功能介绍

获取月份。

2.7.33.2 接口定义

函数接口	uint32_t LL_RTC_DATE_GetMonth (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_MONTH_JANUARY, LL_RTC_MONTH_FEBRUARY, LL_RTC_MONTH_MARCH, LL_RTC_MONTH_APRIL, LL_RTC_MONTH_MAY, LL_RTC_MONTH_JUNE, LL_RTC_MONTH_JULY, LL_RTC_MONTH_AUGUST, LL_RTC_MONTH_SEPTEMBER, LL_RTC_MONTH_OCTOBER, LL_RTC_MONTH_NOVEMBER, LL_RTC_MONTH_DECEMBER }
资源使用	
说明	

2.7.34 LL_RTC_DATE_SetDay

2.7.34.1 功能介绍

设置日。

2.7.34.2 接口定义

函数接口	void LL_RTC_DATE_SetDay (RTC_TypeDef * RTCx, uint32_t Day)
输入	RTC_TypeDef * RTCx uint32_t Day: [0x01, 0x31]
输出	无
返回值	无
资源使用	
说明	

2.7.35 LL_RTC_DATE_GetDay

2.7.35.1 功能介绍

获取日。

2.7.35.2 接口定义

函数接口	uint32_t LL_RTC_DATE_GetDay (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x01, 0x31]
资源使用	
说明	

2.7.36 LL_RTC_DATE_Config

2.7.36.1 功能介绍

设置日期中的年、月、日、星期。

2.7.36.2 接口定义

函数接口	void LL_RTC_DATE_Config (RTC_TypeDef * RTCx, uint32_t WeekDay, uint32_t Day, uint32_t Month, uint32_t Year)
输入	RTC_TypeDef * RTCx uint32_t WeekDay: { LL_RTC_WEEKDAY_MONDAY, LL_RTC_WEEKDAY_TUESDAY, LL_RTC_WEEKDAY_WEDNESDAY, LL_RTC_WEEKDAY_THURSDAY, LL_RTC_WEEKDAY_FRIDAY, LL_RTC_WEEKDAY_SATURDAY, LL_RTC_WEEKDAY_SUNDAY } uint32_t Day: [0x01, 0x31] uint32_t Month: { LL_RTC_MONTH_JANUARY, LL_RTC_MONTH_FEBRUARY, LL_RTC_MONTH_MARCH, LL_RTC_MONTH_APRIL, LL_RTC_MONTH_MAY, LL_RTC_MONTH_JUNE, LL_RTC_MONTH_JULY, LL_RTC_MONTH_AUGUST, LL_RTC_MONTH_SEPTEMBER, LL_RTC_MONTH_OCTOBER, LL_RTC_MONTH_NOVEMBER, LL_RTC_MONTH_DECEMBER } uint32_t Year: [0x00, 0x99]
输出	无
返回值	无
资源使用	
说明	

2.7.37 LL_RTC_DATE_Get

2.7.37.1 功能介绍

获取日期中的年、月、日、星期。

2.7.37.2 接口定义

函数接口	uint32_t LL_RTC_DATE_Get (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx

输出	无
返回值	Format: 0xWWDDMMYY
资源使用	
说明	

2.7.38 LL_RTC_ALM_Enable

2.7.38.1 功能介绍

使能闹钟。

2.7.38.2 接口定义

函数接口	void LL_RTC_ALM_Enable (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.39 LL_RTC_ALM_Disable

2.7.39.1 功能介绍

禁止使能闹钟。

2.7.39.2 接口定义

函数接口	void LL_RTC_ALM_Disable (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.40 LL_RTC_ALM_IsEnabled

2.7.40.1 功能介绍

检查是否使能闹钟。

2.7.40.2 接口定义

函数接口	uint32_t LL_RTC_ALM_IsEnabled (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.41 LL_RTC_ALM_SetWeekDay

2.7.41.1 功能介绍

设置闹钟星期。

2.7.41.2 接口定义

函数接口	void LL_RTC_ALM_SetWeekDay (RTC_TypeDef * RTCx, uint32_t WeekDay)
输入	RTC_TypeDef * RTCx uint32_t WeekDay: { LL_RTC_ALM_WEEKDAY_MONDAY, LL_RTC_ALM_WEEKDAY_TUESDAY, LL_RTC_ALM_WEEKDAY_WEDNESDAY, LL_RTC_ALM_WEEKDAY_THURSDAY, LL_RTC_ALM_WEEKDAY_FRIDAY, LL_RTC_ALM_WEEKDAY_SATURDAY, LL_RTC_ALM_WEEKDAY_SUNDAY }
输出	无
返回值	无
资源使用	
说明	

2.7.42 LL_RTC_ALM_GetWeekDay

2.7.42.1 功能介绍

获取闹钟星期。

2.7.42.2 接口定义

函数接口	uint32_t LL_RTC_ALM_GetWeekDay (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{ LL_RTC_ALM_WEEKDAY_MONDAY, LL_RTC_ALM_WEEKDAY_TUESDAY, LL_RTC_ALM_WEEKDAY_WEDNESDAY, LL_RTC_ALM_WEEKDAY_THURSDAY, LL_RTC_ALM_WEEKDAY_FRIDAY, LL_RTC_ALM_WEEKDAY_SATURDAY, LL_RTC_ALM_WEEKDAY_SUNDAY }
资源使用	
说明	

2.7.43 LL_RTC_ALM_SetHour

2.7.43.1 功能介绍

设置闹钟时钟。

2.7.43.2 接口定义

函数接口	void LL_RTC_ALM_SetHour (RTC_TypeDef * RTCx, uint32_t Hours)
输入	RTC_TypeDef * RTCx uint32_t Hours: [0x01, 0x12] [0x21, 0x32] OR [0x00, 0x23]
输出	无

返回值	无
资源使用	
说明	

2.7.44 LL_RTC_ALM_GetHour

2.7.44.1 功能介绍

获取闹钟时钟。

2.7.44.2 接口定义

函数接口	uint32_t LL_RTC_ALM_GetHour (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x01, 0x12] [0x21, 0x32] OR [0x00, 0x23]
资源使用	
说明	

2.7.45 LL_RTC_ALM_SetMinute

2.7.45.1 功能介绍

设置闹钟分钟。

2.7.45.2 接口定义

函数接口	void LL_RTC_ALM_SetMinute (RTC_TypeDef * RTCx, uint32_t Minutes)
输入	RTC_TypeDef * RTCx uint32_t Minutes: [0x00, 0x59]
输出	无
返回值	无
资源使用	
说明	

2.7.46 LL_RTC_ALM_GetMinute

2.7.46.1 功能介绍

获取闹钟分钟。

2.7.46.2 接口定义

函数接口	uint32_t LL_RTC_ALM_GetMinute (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x00, 0x59]
资源使用	
说明	

2.7.47 LL_RTC_ALM_ConfigTime

2.7.47.1 功能介绍

设置闹钟时间（星期、时、分）。

2.7.47.2 接口定义

函数接口	void LL_RTC_ALM_ConfigTime (RTC_TypeDef * RTCx, uint32_t Hours, uint32_t Minutes, uint32_t WeekDay)
输入	RTC_TypeDef * RTCx uint32_t Hours: [0x01, 0x12] [0x21, 0x32] OR [0x00, 0x23] uint32_t Minutes: [0x00, 0x59] uint32_t WeekDay: { LL_RTC_ALM_WEEKDAY_MONDAY, LL_RTC_ALM_WEEKDAY_TUESDAY, LL_RTC_ALM_WEEKDAY_WEDNESDAY, LL_RTC_ALM_WEEKDAY_THURSDAY, LL_RTC_ALM_WEEKDAY_FRIDAY, LL_RTC_ALM_WEEKDAY_SATURDAY, LL_RTC_ALM_WEEKDAY_SUNDAY }
输出	无
返回值	无
资源使用	
说明	

2.7.48 LL_RTC_SetCompensateValue

2.7.48.1 功能介绍

设置时钟补偿值。

2.7.48.2 接口定义

函数接口	void LL_RTC_SetCompensateValue (RTC_TypeDef * RTCx, uint32_t CompensateValue)
输入	RTC_TypeDef * RTCx uint32_t CompensateValue: [0x00, 0x1FF]
输出	无
返回值	无
资源使用	
说明	

2.7.49 LL_RTC_GetCompensateValue

2.7.49.1 功能介绍

获取时钟补偿值。

2.7.49.2 接口定义

函数接口	uint32_t LL_RTC_GetCompensateValue (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	[0x00, 0x1FF]
资源使用	
说明	

2.7.50 LL_RTC_EnableCompensate

2.7.50.1 功能介绍

使能时钟误差补偿。

2.7.50.2 接口定义

函数接口	void LL_RTC_EnableCompensate (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.51 LL_RTC_DisableCompensate

2.7.51.1 功能介绍

禁止使能时钟误差补偿。

2.7.51.2 接口定义

函数接口	void LL_RTC_DisableCompensate (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.52 LL_RTC_IsEnabledCompensate

2.7.52.1 功能介绍

检查是否使能时钟误差补偿。

2.7.52.2 接口定义

函数接口	uint32_t LL_RTC_IsEnabledCompensate (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.53 LL_RTC_IsActiveFlag_ALM

2.7.53.1 功能介绍

检查是否发生闹钟中断。

2.7.53.2 接口定义

函数接口	uint32_t LL_RTC_IsActiveFlag_ALM (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.54 LL_RTC_IsActiveFlag_PRD

2.7.54.1 功能介绍

检查是否发生周期中断。

2.7.54.2 接口定义

函数接口	uint32_t LL_RTC_IsActiveFlag_PRD (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.55 LL_RTC_ClearFlag_ALM

2.7.55.1 功能介绍

清除闹钟中断标志。

2.7.55.2 接口定义

函数接口	void LL_RTC_ClearFlag_ALM (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.56 LL_RTC_ClearFlag_PRD

2.7.56.1 功能介绍

清除周期中断标志。

2.7.56.2 接口定义

函数接口	void LL_RTC_ClearFlag_PRD (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无

返回值	无
资源使用	
说明	

2.7.57 LL_RTC_ALM_EnableIT

2.7.57.1 功能介绍

使能闹钟中断。

2.7.57.2 接口定义

函数接口	void LL_RTC_ALM_EnableIT (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.58 LL_RTC_ALM_DisableIT

2.7.58.1 功能介绍

禁止使能闹钟中断。

2.7.58.2 接口定义

函数接口	void LL_RTC_ALM_DisableIT (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	无
资源使用	
说明	

2.7.59 LL_RTC_ALM_IsEnabledIT

2.7.59.1 功能介绍

检查是否使能闹钟中断。

2.7.59.2 接口定义

函数接口	uint32_t LL_RTC_ALM_IsEnabledIT (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.7.60 LL_RTC_DeInit

2.7.60.1 功能介绍

清除 RTC 初始化参数。

2.7.60.2 接口定义

函数接口	ErrorStatus LL_RTC_DeInit (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.7.61 LL_RTC_Init

2.7.61.1 功能介绍

RTC 初始化。

2.7.61.2 接口定义

函数接口	ErrorStatus LL_RTC_Init (RTC_TypeDef * RTCx, LL_RTC_InitTypeDef * RTC_InitStruct)
输入	RTC_TypeDef * RTCx LL_RTC_InitTypeDef * RTC_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.7.62 LL_RTC_StructInit

2.7.62.1 功能介绍

RTC 结构体初始化。

2.7.62.2 接口定义

函数接口	void LL_RTC_StructInit (LL_RTC_InitTypeDef * RTC_InitStruct)
输入	LL_RTC_InitTypeDef * RTC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.7.63 LL_RTC_TIME_Init

2.7.63.1 功能介绍

设置 RTC TIME 初始化。

2.7.63.2 接口定义

函数接口	ErrorStatus LL_RTC_TIME_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_TimeTypeDef * RTC_TimeStruct)
输入	RTC_TypeDef * RTCx uint32_t RTC_Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD } LL_RTC_TimeTypeDef * RTC_TimeStruct

输出	无
返回值	ErrorStatus
资源使用	
说明	

2.7.64 LL_RTC_TIME_StructInit

2.7.64.1 功能介绍

设置 RTC TIME 结构体初始化。

2.7.64.2 接口定义

函数接口	void LL_RTC_TIME_StructInit (LL_RTC_TimeTypeDef * RTC_TimeStruct)
输入	LL_RTC_TimeTypeDef * RTC_TimeStruct
输出	无
返回值	无
资源使用	
说明	

2.7.65 LL_RTC_DATE_Init

2.7.65.1 功能介绍

设置 RTC DATE 初始化。

2.7.65.2 接口定义

函数接口	ErrorStatus LL_RTC_DATE_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_DateTypeDef * RTC_DateStruct)
输入	RTC_TypeDef * RTCx uint32_t RTC_Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD } LL_RTC_DateTypeDef * RTC_DateStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.7.66 LL_RTC_DATE_StructInit

2.7.66.1 功能介绍

设置 RTC DATE 结构体初始化。

2.7.66.2 接口定义

函数接口	void LL_RTC_DATE_StructInit (LL_RTC_DateTypeDef * RTC_DateStruct)
输入	LL_RTC_DateTypeDef * RTC_DateStruct
输出	无
返回值	无
资源使用	

说明	
----	--

2.7.67 LL_RTC_ALM_Init

2.7.67.1 功能介绍

设置 RTC ALM 初始化。

2.7.67.2 接口定义

函数接口	ErrorStatus LL_RTC_ALM_Init (RTC_TypeDef * RTCx, uint32_t RTC_Format, LL_RTC_AlarmTypeDef * RTC_AlarmStruct)
输入	RTC_TypeDef * RTCx uint32_t RTC_Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD } LL_RTC_AlarmTypeDef * RTC_AlarmStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.7.68 LL_RTC_ALM_StructInit

2.7.68.1 功能介绍

设置 RTC ALM 结构体初始化。

2.7.68.2 接口定义

函数接口	void LL_RTC_ALM_StructInit (LL_RTC_AlarmTypeDef * RTC_AlarmStruct)
输入	LL_RTC_AlarmTypeDef * RTC_AlarmStruct
输出	无
返回值	无
资源使用	
说明	

2.7.69 LL_RTC_EnterReadWriteMode

2.7.69.1 功能介绍

进入写入/读出模式。

2.7.69.2 接口定义

函数接口	ErrorStatus LL_RTC_EnterReadWriteMode (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.7.70 LL_RTC_ExitReadWriteMode

2.7.70.1 功能介绍

退出写入/读出模式。

2.7.70.2 接口定义

函数接口	ErrorStatus LL_RTC_ExitReadWriteMode (RTC_TypeDef * RTCx)
输入	RTC_TypeDef * RTCx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.8 ADC 模块

属性	类型	字段名	含义
ADC_TypeDef			
读写	uint32_t	CR	控制寄存器
读写	uint32_t	CFG1	配置寄存器 1
读写	uint32_t	CFG2	配置寄存器 2
读写	uint32_t	ISR	中断和状态寄存器
读写	uint32_t	IER	中断使能寄存器
读写	uint32_t	SAMR	采样时间寄存器
读写	uint32_t	CHSELR1	通道选择器寄存器 1
读写	uint32_t	CHSELR2	通道选择器寄存器 2
读写	uint32_t	AWD1TR	看门狗 1 监控电压阈值寄存器
读写	uint32_t	CALFACT	校准系数
只读	uint32_t	DR	数据寄存器
LL_ADC_InitTypeDef			
读写	uint32_t	Clock	时钟源
读写	uint32_t	DataAlignment	数据对齐模式
读写	uint32_t	LowPowerMode	低功耗模式
读写	uint32_t	Vref	参考电压
LL_ADC_REG_InitTypeDef			
读写	uint32_t	TriggerSource	触发源
读写	uint32_t	SequencerLength	序列长度
读写	uint32_t	ConversionMode	转换模式
读写	uint32_t	DMATransfer	DMA 传输
读写	uint32_t	Overrun	溢出

2.8.1 LL_ADC_DMA_GetRegAddr

2.8.1.1 功能介绍

获取 ADC 寄存器地址。

2.8.1.2 接口定义

函数接口	uint32_t LL_ADC_DMA_GetRegAddr (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	ADC 寄存器地址
资源使用	
说明	

2.8.2 LL_ADC_SetClock

2.8.2.1 功能介绍

设置 ADC 时钟源和预分频值。

2.8.2.2 接口定义

函数接口	void LL_ADC_SetClock (ADC_TypeDef * ADCx, uint32_t ClockSource)
输入	ADC_TypeDef * ADCx uint32_t ClockSource: { LL_ADC_CLOCK_SYNC_PCLK_DIV1, LL_ADC_CLOCK_SYNC_PCLK_DIV2, LL_ADC_CLOCK_SYNC_PCLK_DIV4, LL_ADC_CLOCK_ASYNC_DIV1, LL_ADC_CLOCK_ASYNC_DIV2, LL_ADC_CLOCK_ASYNC_DIV4, LL_ADC_CLOCK_ASYNC_DIV6, LL_ADC_CLOCK_ASYNC_DIV8, LL_ADC_CLOCK_ASYNC_DIV10, LL_ADC_CLOCK_ASYNC_DIV12, LL_ADC_CLOCK_ASYNC_DIV16, LL_ADC_CLOCK_ASYNC_DIV32, LL_ADC_CLOCK_ASYNC_DIV64, LL_ADC_CLOCK_ASYNC_DIV128, LL_ADC_CLOCK_ASYNC_DIV256 }
输出	无
返回值	无
资源使用	
说明	

2.8.3 LL_ADC_GetClock

2.8.3.1 功能介绍

获取 ADC 时钟源和预分频值。

2.8.3.2 接口定义

函数接口	uint32_t LL_ADC_GetClock (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无

返回值	{ LL_ADC_CLOCK_SYNC_PCLK_DIV1, LL_ADC_CLOCK_SYNC_PCLK_DIV2, LL_ADC_CLOCK_SYNC_PCLK_DIV4, LL_ADC_CLOCK_ASYNC_DIV1, LL_ADC_CLOCK_ASYNC_DIV2, LL_ADC_CLOCK_ASYNC_DIV4, LL_ADC_CLOCK_ASYNC_DIV6, LL_ADC_CLOCK_ASYNC_DIV8, LL_ADC_CLOCK_ASYNC_DIV10, LL_ADC_CLOCK_ASYNC_DIV12, LL_ADC_CLOCK_ASYNC_DIV16, LL_ADC_CLOCK_ASYNC_DIV32, LL_ADC_CLOCK_ASYNC_DIV64, LL_ADC_CLOCK_ASYNC_DIV128, LL_ADC_CLOCK_ASYNC_DIV256 }
资源使用	
说明	

2.8.4 LL_ADC_SetPathInternalCh

2.8.4.1 功能介绍

设置内部通道的路径。

2.8.4.2 接口定义

函数接口	void LL_ADC_SetPathInternalCh (ADC_TypeDef * ADCx, uint32_t PathInternal)
输入	ADC_TypeDef * ADCx uint32_t PathInternal: { LL_ADC_PATH_INTERNAL_NONE, LL_ADC_PATH_INTERNAL_VREFINT, LL_ADC_PATH_INTERNAL_TEMPSENSOR, LL_ADC_PATH_INTERNAL_VCC }
输出	无
返回值	无
资源使用	
说明	

2.8.5 LL_ADC_GetPathInternalCh

2.8.5.1 功能介绍

获取内部通道的路径。

2.8.5.2 接口定义

函数接口	uint32_t LL_ADC_GetPathInternalCh (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_PATH_INTERNAL_NONE, LL_ADC_PATH_INTERNAL_VREFINT, LL_ADC_PATH_INTERNAL_TEMPSENSOR, LL_ADC_PATH_INTERNAL_VCC }

资源使用	
说明	

2.8.6 LL_ADC_EnableChBuf

2.8.6.1 功能介绍

使能 ADC 通道缓冲区。

2.8.6.2 接口定义

函数接口	void LL_ADC_EnableChBuf (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.7 LL_ADC_DisableChBuf

2.8.7.1 功能介绍

禁止使能 ADC 通道缓冲区。

2.8.7.2 接口定义

函数接口	void LL_ADC_DisableChBuf (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.8 LL_ADC_IsEnabledChBuf

2.8.8.1 功能介绍

检查是否使能 ADC 通道缓冲区。

2.8.8.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledChBuf (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.9 LL_ADC_SetVREF

2.8.9.1 功能介绍

设置 ADC 参考电压。

2.8.9.2 接口定义

函数接口	void LL_ADC_SetVREF (ADC_TypeDef * ADCx, uint32_t VoltageRef)
------	---

输入	ADC_TypeDef * ADCx uint32_t VoltageRef: { LL_ADC_VREF_1V25, LL_ADC_VREF_2V5, LL_ADC_VREF_3V, LL_ADC_VREF_4V, LL_ADC_VREF_VCC, LL_ADC_VREF_PIN }
输出	无
返回值	无
资源使用	
说明	

2.8.10 LL_ADC_GetVREF

2.8.10.1 功能介绍

获取 ADC 参考电压。

2.8.10.2 接口定义

函数接口	uint32_t LL_ADC_GetVREF (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_VREF_1V25, LL_ADC_VREF_2V5, LL_ADC_VREF_3V, LL_ADC_VREF_4V, LL_ADC_VREF_VCC, LL_ADC_VREF_PIN }
资源使用	
说明	

2.8.11 LL_ADC_SetCalibrationFactor

2.8.11.1 功能介绍

设置 ADC 校准因子。

2.8.11.2 接口定义

函数接口	void LL_ADC_SetCalibrationFactor (ADC_TypeDef * ADCx, uint32_t CalibrationFactor)
输入	ADC_TypeDef * ADCx uint32_t CalibrationFactor: [0x00, 0x3F]
输出	无
返回值	无
资源使用	
说明	

2.8.12 LL_ADC_GetCalibrationFactor

2.8.12.1 功能介绍

获取 ADC 校准因子。

2.8.12.2 接口定义

函数接口	uint32_t LL_ADC_GetCalibrationFactor (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无

返回值	[0x00, 0x3F]
资源使用	
说明	

2.8.13 LL_ADC_SetCalibrationGainK

2.8.13.1 功能介绍

设置 ADC 校准因子增益值 k。

2.8.13.2 接口定义

函数接口	void LL_ADC_SetCalibrationGainK (ADC_TypeDef * ADCx, uint32_t GainK)
输入	ADC_TypeDef * ADCx uint32_t GainK: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.8.14 LL_ADC_GetCalibrationGainK

2.8.14.1 功能介绍

获取 ADC 校准因子增益值 k。

2.8.14.2 接口定义

函数接口	uint32_t LL_ADC_GetCalibrationGainK (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.8.15 LL_ADC_SetCalibrationGainB

2.8.15.1 功能介绍

设置 ADC 校准因子增益值 B。

2.8.15.2 接口定义

函数接口	void LL_ADC_SetCalibrationGainB (ADC_TypeDef * ADCx, uint32_t GainB)
输入	ADC_TypeDef * ADCx uint32_t GainB: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.8.16 LL_ADC_GetCalibrationGainB

2.8.16.1 功能介绍

获取 ADC 校准因子增益值 B。

2.8.16.2 接口定义

函数接口	uint32_t LL_ADC_GetCalibrationGainB (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.8.17 LL_ADC_SetCalibrationTimes

2.8.17.1 功能介绍

设置 ADC 校准因子增益倍数。

2.8.17.2 接口定义

函数接口	void LL_ADC_SetCalibrationTimes (ADC_TypeDef * ADCx, uint32_t Times)
输入	ADC_TypeDef * ADCx uint32_t Times: { LL_ADC_CALTIMES_4, LL_ADC_CALTIMES_8, LL_ADC_CALTIMES_16, LL_ADC_CALTIMES_64 }
输出	无
返回值	无
资源使用	
说明	

2.8.18 LL_ADC_GetCalibrationTimes

2.8.18.1 功能介绍

获取 ADC 校准因子增益倍数。

2.8.18.2 接口定义

函数接口	uint32_t LL_ADC_GetCalibrationTimes (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_CALTIMES_4, LL_ADC_CALTIMES_8, LL_ADC_CALTIMES_16, LL_ADC_CALTIMES_64 }
资源使用	
说明	

2.8.19 LL_ADC_SetDataAlignment

2.8.19.1 功能介绍

设置 ADC 转换数据对齐方式。

2.8.19.2 接口定义

函数接口	void LL_ADC_SetDataAlignment (ADC_TypeDef * ADCx, uint32_t DataAlignment)
输入	ADC_TypeDef * ADCx uint32_t DataAlignment: { LL_ADC_DATA_ALIGN_RIGHT, LL_ADC_DATA_ALIGN_LEFT }
输出	无
返回值	无
资源使用	
说明	

2.8.20 LL_ADC_GetDataAlignment

2.8.20.1 功能介绍

获取 ADC 转换数据对齐方式。

2.8.20.2 接口定义

函数接口	uint32_t LL_ADC_GetDataAlignment (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_DATA_ALIGN_RIGHT, LL_ADC_DATA_ALIGN_LEFT }
资源使用	
说明	

2.8.21 LL_ADC_SetLowPowerMode

2.8.21.1 功能介绍

设置 ADC 低功耗模式。

2.8.21.2 接口定义

函数接口	void LL_ADC_SetLowPowerMode (ADC_TypeDef * ADCx, uint32_t LowPowerMode)
输入	ADC_TypeDef * ADCx uint32_t LowPowerMode: { LL_ADC_LP_MODE_NONE, LL_ADC_LP_MODE_AUTOWAIT }
输出	无
返回值	无
资源使用	
说明	

2.8.22 LL_ADC_GetLowPowerMode

2.8.22.1 功能介绍

获取 ADC 低功耗模式。

2.8.22.2 接口定义

函数接口	uint32_t LL_ADC_GetLowPowerMode (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx

输出	无
返回值	{ LL_ADC_LP_MODE_NONE, LL_ADC_LP_MODE_AUTOWAIT }
资源使用	
说明	

2.8.23 LL_ADC_REG_SetTriggerSource

2.8.23.1 功能介绍

设置 ADC 触发源。

2.8.23.2 接口定义

函数接口	void LL_ADC_REG_SetTriggerSource (ADC_TypeDef * ADCx, uint32_t TriggerSource)
输入	ADC_TypeDef * ADCx uint32_t TriggerSource: { LL_ADC_REG_TRIG_SOFTWARE, LL_ADC_REG_TRIG_EXT_TIM1_TRGO2, LL_ADC_REG_TRIG_EXT_TIM1_CH4, LL_ADC_REG_TRIG_EXT_TIM2_TRGO, LL_ADC_REG_TRIG_EXT_PLA_OUT1, LL_ADC_REG_TRIG_EXT_TIM4_TRGO, LL_ADC_REG_TRIG_EXT_PLA_OUT2, LL_ADC_REG_TRIG_EXT_TIM6_TRGO, LL_ADC_REG_TRIG_EXT_EXTI_LINE11 }
输出	无
返回值	无
资源使用	
说明	

2.8.24 LL_ADC_REG_GetTriggerSource

2.8.24.1 功能介绍

获取 ADC 触发源。

2.8.24.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetTriggerSource (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_REG_TRIG_SOFTWARE, LL_ADC_REG_TRIG_EXT_TIM1_TRGO2, LL_ADC_REG_TRIG_EXT_TIM1_CH4, LL_ADC_REG_TRIG_EXT_TIM2_TRGO, LL_ADC_REG_TRIG_EXT_PLA_OUT1, LL_ADC_REG_TRIG_EXT_TIM4_TRGO, LL_ADC_REG_TRIG_EXT_PLA_OUT2, LL_ADC_REG_TRIG_EXT_TIM6_TRGO, LL_ADC_REG_TRIG_EXT_EXTI_LINE11 }

资源使用	
说明	

2.8.25 LL_ADC_REG_SetTriggerEdge

2.8.25.1 功能介绍

设置 ADC 外部触发使能和极性选择。

2.8.25.2 接口定义

函数接口	void LL_ADC_REG_SetTriggerEdge (ADC_TypeDef * ADCx, uint32_t ExternalTriggerEdge)
输入	ADC_TypeDef * ADCx uint32_t ExternalTriggerEdge: { LL_ADC_REG_TRIG_EXT_NONE, LL_ADC_REG_TRIG_EXT_RISING, LL_ADC_REG_TRIG_EXT_FALLING, LL_ADC_REG_TRIG_EXT_RISINGFALLING }
输出	无
返回值	无
资源使用	
说明	

2.8.26 LL_ADC_REG_GetTriggerEdge

2.8.26.1 功能介绍

获取 ADC 外部触发使能和极性选择。

2.8.26.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetTriggerEdge (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_REG_TRIG_EXT_NONE, LL_ADC_REG_TRIG_EXT_RISING, LL_ADC_REG_TRIG_EXT_FALLING, LL_ADC_REG_TRIG_EXT_RISINGFALLING }
资源使用	
说明	

2.8.27 LL_ADC_REG_SetSequencerLength

2.8.27.1 功能介绍

设置 ADC 序列长度。

2.8.27.2 接口定义

函数接口	void LL_ADC_REG_SetSequencerLength (ADC_TypeDef * ADCx, uint32_t SequencerNbRanks)
输入	ADC_TypeDef * ADCx uint32_t SequencerNbRanks: { LL_ADC_REG_SEQ_SCAN_DISABLE,

	LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS }
输出	无
返回值	无
资源使用	
说明	

2.8.28 LL_ADC_REG_GetSequencerLength

2.8.28.1 功能介绍

获取 ADC 序列长度。

2.8.28.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetSequencerLength (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_REG_SEQ_SCAN_DISABLE, LL_ADC_REG_SEQ_SCAN_ENABLE_2RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_3RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_4RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_5RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_6RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_7RANKS, LL_ADC_REG_SEQ_SCAN_ENABLE_8RANKS }
资源使用	
说明	

2.8.29 LL_ADC_REG_SetSequencerRanks

2.8.29.1 功能介绍

设置 ADC 通道序列转换次序。

2.8.29.2 接口定义

函数接口	void LL_ADC_REG_SetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank, uint32_t Channel)
输入	ADC_TypeDef * ADCx uint32_t Rank: { LL_ADC_REG_RANK_1, LL_ADC_REG_RANK_2, LL_ADC_REG_RANK_3, LL_ADC_REG_RANK_4, LL_ADC_REG_RANK_5, LL_ADC_REG_RANK_6, LL_ADC_REG_RANK_7, LL_ADC_REG_RANK_8 } uint32_t Channel:

	<pre>{ LL_ADC_CHANNEL_0, LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, LL_ADC_CHANNEL_3, LL_ADC_CHANNEL_4, LL_ADC_CHANNEL_5, LL_ADC_CHANNEL_6, LL_ADC_CHANNEL_7, LL_ADC_CHANNEL_8, LL_ADC_CHANNEL_9, LL_ADC_CHANNEL_10, LL_ADC_CHANNEL_11, LL_ADC_CHANNEL_12, LL_ADC_CHANNEL_13, LL_ADC_CHANNEL_14, LL_ADC_CHANNEL_15, LL_ADC_CHANNEL_16, LL_ADC_CHANNEL_17, LL_ADC_CHANNEL_18, LL_ADC_CHANNEL_19, LL_ADC_CHANNEL_20, LL_ADC_CHANNEL_21, LL_ADC_CHANNEL_22, LL_ADC_CHANNEL_23, LL_ADC_CHANNEL_24, LL_ADC_CHANNEL_25, LL_ADC_CHANNEL_26, LL_ADC_CHANNEL_27, LL_ADC_CHANNEL_28, LL_ADC_CHANNEL_29, LL_ADC_CHANNEL_30, LL_ADC_CHANNEL_31, LL_ADC_CHANNEL_32, LL_ADC_CHANNEL_33, LL_ADC_CHANNEL_34, LL_ADC_CHANNEL_35, LL_ADC_CHANNEL_36, LL_ADC_CHANNEL_37, LL_ADC_CHANNEL_38, LL_ADC_CHANNEL_39, LL_ADC_CHANNEL_40, LL_ADC_CHANNEL_41, LL_ADC_CHANNEL_42, LL_ADC_CHANNEL_43, LL_ADC_CHANNEL_44, LL_ADC_CHANNEL_45, LL_ADC_CHANNEL_46, LL_ADC_CHANNEL_47, LL_ADC_CHANNEL_48, LL_ADC_CHANNEL_49, LL_ADC_CHANNEL_50, LL_ADC_CHANNEL_51, LL_ADC_CHANNEL_52, LL_ADC_CHANNEL_53, LL_ADC_CHANNEL_54, LL_ADC_CHANNEL_55, LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, LL_ADC_CHANNEL_VCC }</pre>
输出	无
返回值	无
资源使用	
说明	

2.8.30 LL_ADC_REG_GetSequencerRanks

2.8.30.1 功能介绍

获取 ADC 通道序列转换次序。

2.8.30.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetSequencerRanks (ADC_TypeDef * ADCx, uint32_t Rank)
输入	ADC_TypeDef * ADCx uint32_t Rank: { LL_ADC_REG_RANK_1, LL_ADC_REG_RANK_2,

	LL_ADC_REG_RANK_3, LL_ADC_REG_RANK_4, LL_ADC_REG_RANK_5, LL_ADC_REG_RANK_6, LL_ADC_REG_RANK_7, LL_ADC_REG_RANK_8 }
输出	无
返回值	{ LL_ADC_CHANNEL_0, LL_ADC_CHANNEL_1, LL_ADC_CHANNEL_2, LL_ADC_CHANNEL_3, LL_ADC_CHANNEL_4, LL_ADC_CHANNEL_5, LL_ADC_CHANNEL_6, LL_ADC_CHANNEL_7, LL_ADC_CHANNEL_8, LL_ADC_CHANNEL_9, LL_ADC_CHANNEL_10, LL_ADC_CHANNEL_11, LL_ADC_CHANNEL_12, LL_ADC_CHANNEL_13, LL_ADC_CHANNEL_14, LL_ADC_CHANNEL_15, LL_ADC_CHANNEL_16, LL_ADC_CHANNEL_17, LL_ADC_CHANNEL_18, LL_ADC_CHANNEL_19, LL_ADC_CHANNEL_20, LL_ADC_CHANNEL_21, LL_ADC_CHANNEL_22, LL_ADC_CHANNEL_23, LL_ADC_CHANNEL_24, LL_ADC_CHANNEL_25, LL_ADC_CHANNEL_26, LL_ADC_CHANNEL_27, LL_ADC_CHANNEL_28, LL_ADC_CHANNEL_29, LL_ADC_CHANNEL_30, LL_ADC_CHANNEL_31, LL_ADC_CHANNEL_32, LL_ADC_CHANNEL_33, LL_ADC_CHANNEL_34, LL_ADC_CHANNEL_35, LL_ADC_CHANNEL_36, LL_ADC_CHANNEL_37, LL_ADC_CHANNEL_38, LL_ADC_CHANNEL_39, LL_ADC_CHANNEL_40, LL_ADC_CHANNEL_41, LL_ADC_CHANNEL_42, LL_ADC_CHANNEL_43, LL_ADC_CHANNEL_44, LL_ADC_CHANNEL_45, LL_ADC_CHANNEL_46, LL_ADC_CHANNEL_47, LL_ADC_CHANNEL_48, LL_ADC_CHANNEL_49, LL_ADC_CHANNEL_50, LL_ADC_CHANNEL_51, LL_ADC_CHANNEL_52, LL_ADC_CHANNEL_53, LL_ADC_CHANNEL_54, LL_ADC_CHANNEL_55, LL_ADC_CHANNEL_VREFINT, LL_ADC_CHANNEL_TEMPSENSOR, LL_ADC_CHANNEL_VCC }
资源使用	
说明	

2.8.31 LL_ADC_REG_SetConversionMode

2.8.31.1 功能介绍

设置 ADC 转换模式。

2.8.31.2 接口定义

函数接口	void LL_ADC_REG_SetConversionMode (ADC_TypeDef * ADCx, uint32_t ConversionMode)
输入	ADC_TypeDef * ADCx uint32_t ConversionMode:

	{ LL_ADC_REG_CONV_SINGLE, LL_ADC_REG_CONV_CONTINUOUS, LL_ADC_REG_CONV_DISCONTINUOUS }
输出	无
返回值	无
资源使用	
说明	

2.8.32 LL_ADC_REG_GetConversionMode

2.8.32.1 功能介绍

获取 ADC 转换模式。

2.8.32.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetConversionMode (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_REG_CONV_SINGLE, LL_ADC_REG_CONV_CONTINUOUS, LL_ADC_REG_CONV_DISCONTINUOUS }
资源使用	
说明	

2.8.33 LL_ADC_REG_SetDMATransfer

2.8.33.1 功能介绍

设置 ADC 转换是否使能 DMA。

2.8.33.2 接口定义

函数接口	void LL_ADC_REG_SetDMATransfer (ADC_TypeDef * ADCx, uint32_t DMATransfer)
输入	ADC_TypeDef * ADCx uint32_t DMATransfer: { LL_ADC_REG_DMA_TRANSFER_DISABLE, LL_ADC_REG_DMA_TRANSFER_ENABLE }
输出	无
返回值	无
资源使用	
说明	

2.8.34 LL_ADC_REG_GetDMATransfer

2.8.34.1 功能介绍

获取 ADC 转换是否使能 DMA。

2.8.34.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetDMATransfer (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx

输出	无
返回值	{ LL_ADC_REG_DMA_TRANSFER_DISABLE, LL_ADC_REG_DMA_TRANSFER_ENABLE }
资源使用	
说明	

2.8.35 LL_ADC_REG_SetOverrun

2.8.35.1 功能介绍

设置 ADC 数据溢出管理模式。

2.8.35.2 接口定义

函数接口	void LL_ADC_REG_SetOverrun (ADC_TypeDef * ADCx, uint32_t Overrun)
输入	ADC_TypeDef * ADCx uint32_t Overrun: { LL_ADC_REG_OVR_DATA_PRESERVED, LL_ADC_REG_OVR_DATA_OVERWRITTEN }
输出	无
返回值	无
资源使用	
说明	

2.8.36 LL_ADC_REG_GetOverrun

2.8.36.1 功能介绍

获取 ADC 数据溢出管理模式。

2.8.36.2 接口定义

函数接口	uint32_t LL_ADC_REG_GetOverrun (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_REG_OVR_DATA_PRESERVED, LL_ADC_REG_OVR_DATA_OVERWRITTEN }
资源使用	
说明	

2.8.37 LL_ADC_SetChannelSamplingTime

2.8.37.1 功能介绍

设置 ADC 通道采样时间。

2.8.37.2 接口定义

函数接口	void LL_ADC_SetChannelSamplingTime (ADC_TypeDef * ADCx, uint32_t SamplingTime)
输入	ADC_TypeDef * ADCx uint32_t SamplingTime: { LL_ADC_SAMPLINGTIME_1CYCLE_5, LL_ADC_SAMPLINGTIME_3CYCLES_5,

	LL_ADC_SAMPLINGTIME_7CYCLES_5, LL_ADC_SAMPLINGTIME_12CYCLES_5, LL_ADC_SAMPLINGTIME_19CYCLES_5, LL_ADC_SAMPLINGTIME_39CYCLES_5, LL_ADC_SAMPLINGTIME_79CYCLES_5, LL_ADC_SAMPLINGTIME_119CYCLES_5, LL_ADC_SAMPLINGTIME_159CYCLES_5, LL_ADC_SAMPLINGTIME_199CYCLES_5, LL_ADC_SAMPLINGTIME_239CYCLES_5 }
输出	无
返回值	无
资源使用	
说明	

2.8.38 LL_ADC_GetChannelSamplingTime

2.8.38.1 功能介绍

获取 ADC 通道采样时间。

2.8.38.2 接口定义

函数接口	uint32_t LL_ADC_GetChannelSamplingTime (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{ LL_ADC_SAMPLINGTIME_1CYCLE_5, LL_ADC_SAMPLINGTIME_3CYCLES_5, LL_ADC_SAMPLINGTIME_7CYCLES_5, LL_ADC_SAMPLINGTIME_12CYCLES_5, LL_ADC_SAMPLINGTIME_19CYCLES_5, LL_ADC_SAMPLINGTIME_39CYCLES_5, LL_ADC_SAMPLINGTIME_79CYCLES_5, LL_ADC_SAMPLINGTIME_119CYCLES_5, LL_ADC_SAMPLINGTIME_159CYCLES_5, LL_ADC_SAMPLINGTIME_199CYCLES_5, LL_ADC_SAMPLINGTIME_239CYCLES_5 }
资源使用	
说明	

2.8.39 LL_ADC_AnalogWDG_Enable

2.8.39.1 功能介绍

ADC 当前转换通道模拟看门狗功能使能。

2.8.39.2 接口定义

函数接口	void LL_ADC_AnalogWDG_Enable (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无

资源使用	
说明	

2.8.40 LL_ADC_AnalogWDG_Disable

2.8.40.1 功能介绍

禁止 ADC 当前转换通道模拟看门狗功能使能。

2.8.40.2 接口定义

函数接口	void LL_ADC_AnalogWDG_Disable (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.41 LL_ADC_AnalogWDG_IsEnabled

2.8.41.1 功能介绍

检查 ADC 当前转换通道模拟看门狗功能是否使能。

2.8.41.2 接口定义

函数接口	uint32_t LL_ADC_AnalogWDG_IsEnabled (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.42 LL_ADC_ConfigAnalogWDThresholds

2.8.42.1 功能介绍

设置 ADC 模拟看门狗阈值的高阈值和低阈值。

2.8.42.2 接口定义

函数接口	void LL_ADC_ConfigAnalogWDThresholds (ADC_TypeDef * ADCx, uint32_t AWDThresholdHighValue, uint32_t AWDThresholdLowValue)
输入	ADC_TypeDef * ADCx uint32_t AWDThresholdHighValue: [0x000, 0xFFF] uint32_t AWDThresholdLowValue: [0x000, 0xFFF]
输出	无
返回值	无
资源使用	
说明	

2.8.43 LL_ADC_SetAnalogWDThresholds

2.8.43.1 功能介绍

设置 ADC 模拟看门狗阈值的高阈值或低阈值。

2.8.43.2 接口定义

函数接口	void LL_ADC_SetAnalogWDThresholds (ADC_TypeDef * ADCx, uint32_t AWDThresholdsHighLow, uint32_t AWDThresholdValue)
输入	ADC_TypeDef * ADCx uint32_t AWDThresholdsHighLow: { LL_ADC_AWD_THRESHOLD_HIGH, LL_ADC_AWD_THRESHOLD_LOW } uint32_t AWDThresholdValue: [0x000, 0xFFF]
输出	无
返回值	无
资源使用	
说明	

2.8.44 LL_ADC_GetAnalogWDThresholds

2.8.44.1 功能介绍

获取 ADC 模拟看门狗阈值的高阈值或低阈值。

2.8.44.2 接口定义

函数接口	uint32_t LL_ADC_GetAnalogWDThresholds (ADC_TypeDef * ADCx, uint32_t AWDThresholdsHighLow)
输入	ADC_TypeDef * ADCx uint32_t AWDThresholdsHighLow: { LL_ADC_AWD_THRESHOLD_HIGH, LL_ADC_AWD_THRESHOLD_LOW }
输出	无
返回值	[0x000, 0xFFF]
资源使用	
说明	

2.8.45 LL_ADC_Enable

2.8.45.1 功能介绍

使能 ADC。

2.8.45.2 接口定义

函数接口	void LL_ADC_Enable (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	

说明	
----	--

2.8.46 LL_ADC_Disable

2.8.46.1 功能介绍

禁止使能 ADC。

2.8.46.2 接口定义

函数接口	void LL_ADC_Disable (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.47 LL_ADC_IsEnabled

2.8.47.1 功能介绍

检查是否使能 ADC。

2.8.47.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabled (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.48 LL_ADC_IsDisableOngoing

2.8.48.1 功能介绍

获取 ADC 禁用状态。

2.8.48.2 接口定义

函数接口	uint32_t LL_ADC_IsDisableOngoing (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.49 LL_ADC_StartCalibration

2.8.49.1 功能介绍

开启 ADC 校准。

2.8.49.2 接口定义

函数接口	void LL_ADC_StartCalibration (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx

输出	无
返回值	无
资源使用	
说明	

2.8.50 LL_ADC_IsCalibrationOnGoing

2.8.50.1 功能介绍

获取 ADC 校准状态。

2.8.50.2 接口定义

函数接口	uint32_t LL_ADC_IsCalibrationOnGoing (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.51 LL_ADC_REG_StartConversion

2.8.51.1 功能介绍

开启 ADC 规则组转换。

2.8.51.2 接口定义

函数接口	void LL_ADC_REG_StartConversion (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.52 LL_ADC_REG_StopConversion

2.8.52.1 功能介绍

停止 ADC 规则组转换。

2.8.52.2 接口定义

函数接口	void LL_ADC_REG_StopConversion (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.53 LL_ADC_REG_IsConversionOngoing

2.8.53.1 功能介绍

获取 ADC 规则组转换状态。

2.8.53.2 接口定义

函数接口	uint32_t LL_ADC_REG_IsConversionOngoing (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.54 LL_ADC_REG_IsStopConversionOngoing

2.8.54.1 功能介绍

获取 ADC 规则组停止转换状态。

2.8.54.2 接口定义

函数接口	uint32_t LL_ADC_REG_IsStopConversionOngoing (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.55 LL_ADC_REG_ReadConversionData32

2.8.55.1 功能介绍

获取 ADC 组常规转换 32 位数据。

2.8.55.2 接口定义

函数接口	uint32_t LL_ADC_REG_ReadConversionData32 (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.8.56 LL_ADC_REG_ReadConversionData12

2.8.56.1 功能介绍

获取 ADC 组常规转换 12 位数据。

2.8.56.2 接口定义

函数接口	uint16_t LL_ADC_REG_ReadConversionData12 (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	[0x000, 0xFFF]

资源使用	
说明	

2.8.57 LL_ADC_IsActiveFlag_ADRDY

2.8.57.1 功能介绍

ADC 是否就绪。

2.8.57.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag_ADRDY (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.58 LL_ADC_IsActiveFlag_EOC

2.8.58.1 功能介绍

ADC 是否转换结束。

2.8.58.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag_EOC (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.59 LL_ADC_IsActiveFlag_EOS

2.8.59.1 功能介绍

ADC 是否转换序列已完成。

2.8.59.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag_EOS (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.60 LL_ADC_IsActiveFlag_OVR

2.8.60.1 功能介绍

ADC 是否溢出。

2.8.60.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag_OVR (ADC_TypeDef * ADCx)
------	---

输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.61 LL_ADC_IsActiveFlag_AWDG

2.8.61.1 功能介绍

ADC 是否发生模拟看门狗事件。

2.8.61.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag_AWDG (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.62 LL_ADC_IsActiveFlag_EOCAL

2.8.62.1 功能介绍

ADC 是否校准结束。

2.8.62.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag_EOCAL (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.63 LL_ADC_IsActiveFlag

2.8.63.1 功能介绍

ADC 标志是否发生。

2.8.63.2 接口定义

函数接口	uint32_t LL_ADC_IsActiveFlag (ADC_TypeDef * ADCx, uint32_t FlagType)
输入	ADC_TypeDef * ADCx uint32_t FlagType: { LL_ADC_FLAG_ADRDY, LL_ADC_FLAG_EOC, LL_ADC_FLAG_EOS, LL_ADC_FLAG_OVR, LL_ADC_FLAG_AWD1, LL_ADC_FLAG_EOCAL }
输出	无
返回值	{0,1}

资源使用	
说明	

2.8.64 LL_ADC_ClearFlag_ADRDY

2.8.64.1 功能介绍

清除 ADC 就绪标志位。

2.8.64.2 接口定义

函数接口	void LL_ADC_ClearFlag_ADRDY (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.65 LL_ADC_ClearFlag_EOC

2.8.65.1 功能介绍

清除 ADC 转换结束标志位。

2.8.65.2 接口定义

函数接口	void LL_ADC_ClearFlag_EOC (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.66 LL_ADC_ClearFlag_EOS

2.8.66.1 功能介绍

清除 ADC 转换序列结束标志位。

2.8.66.2 接口定义

函数接口	void LL_ADC_ClearFlag_EOS (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.67 LL_ADC_ClearFlag_OVR

2.8.67.1 功能介绍

清除 ADC 溢出标志位。

2.8.67.2 接口定义

函数接口	void LL_ADC_ClearFlag_OVR (ADC_TypeDef * ADCx)
------	--

输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.68 LL_ADC_ClearFlag_AWD1

2.8.68.1 功能介绍

清除 ADC 看门狗事件标志位。

2.8.68.2 接口定义

函数接口	void LL_ADC_ClearFlag_AWD1 (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.69 LL_ADC_ClearFlag_EOCAL

2.8.69.1 功能介绍

清除 ADC 校准结束标志位。

2.8.69.2 接口定义

函数接口	void LL_ADC_ClearFlag_EOCAL (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.70 LL_ADC_ClearFlag

2.8.70.1 功能介绍

清除标志位。

2.8.70.2 接口定义

函数接口	void LL_ADC_ClearFlag (ADC_TypeDef * ADCx, uint32_t FlagType)
输入	ADC_TypeDef * ADCx uint32_t FlagType: { LL_ADC_FLAG_ADRDY, LL_ADC_FLAG_EOC, LL_ADC_FLAG_EOS, LL_ADC_FLAG_OVR, LL_ADC_FLAG_AWD1, LL_ADC_FLAG_EOCAL }
输出	无
返回值	无
资源使用	

说明	
----	--

2.8.71 LL_ADC_EnableIT_ADRDY

2.8.71.1 功能介绍

使能 ADC 就绪中断。

2.8.71.2 接口定义

函数接口	void LL_ADC_EnableIT_ADRDY (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.72 LL_ADC_EnableIT_EOC

2.8.72.1 功能介绍

使能转换结束中断。

2.8.72.2 接口定义

函数接口	void LL_ADC_EnableIT_EOC (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.73 LL_ADC_EnableIT_EOS

2.8.73.1 功能介绍

使能转换序列结束中断。

2.8.73.2 接口定义

函数接口	void LL_ADC_EnableIT_EOS (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.74 LL_ADC_EnableIT_OVR

2.8.74.1 功能介绍

使能溢出中断。

2.8.74.2 接口定义

函数接口	void LL_ADC_EnableIT_OVR (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx

输出	无
返回值	无
资源使用	
说明	

2.8.75 LL_ADC_EnableIT_AWD1

2.8.75.1 功能介绍

使能模拟看门狗中断。

2.8.75.2 接口定义

函数接口	void LL_ADC_EnableIT_AWD1 (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.76 LL_ADC_EnableIT_EOCAL

2.8.76.1 功能介绍

使能校准结束中断。

2.8.76.2 接口定义

函数接口	void LL_ADC_EnableIT_EOCAL (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.77 LL_ADC_EnableIT

2.8.77.1 功能介绍

使能 ADC 中断。

2.8.77.2 接口定义

函数接口	void LL_ADC_EnableIT (ADC_TypeDef * ADCx, uint32_t Interrupt)
输入	ADC_TypeDef * ADCx uint32_t Interrupt: { LL_ADC_IT_ADRDY, LL_ADC_IT_EOC, LL_ADC_IT_EOS, LL_ADC_IT_OVR, LL_ADC_IT_AWD1, LL_ADC_IT_EOCAL }
输出	无
返回值	无
资源使用	
说明	

2.8.78 LL_ADC_DisableIT_ADRDY

2.8.78.1 功能介绍

禁止使能 ADC 就绪中断。

2.8.78.2 接口定义

函数接口	void LL_ADC_DisableIT_ADRDY (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.79 LL_ADC_DisableIT_EOC

2.8.79.1 功能介绍

禁止使能转换结束中断。

2.8.79.2 接口定义

函数接口	void LL_ADC_DisableIT_EOC (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.80 LL_ADC_DisableIT_EOS

2.8.80.1 功能介绍

禁止使能转换序列结束中断。

2.8.80.2 接口定义

函数接口	void LL_ADC_DisableIT_EOS (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.81 LL_ADC_DisableIT_OVR

2.8.81.1 功能介绍

禁止使能溢出中断。

2.8.81.2 接口定义

函数接口	void LL_ADC_DisableIT_OVR (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无

返回值	无
资源使用	
说明	

2.8.82 LL_ADC_DisableIT_AWD1

2.8.82.1 功能介绍

禁止使能模拟看门狗中断。

2.8.82.2 接口定义

函数接口	void LL_ADC_DisableIT_AWD1 (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.83 LL_ADC_DisableIT_EOCAL

2.8.83.1 功能介绍

禁止使能校准结束中断。

2.8.83.2 接口定义

函数接口	void LL_ADC_DisableIT_EOCAL (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	无
资源使用	
说明	

2.8.84 LL_ADC_DisableIT

2.8.84.1 功能介绍

禁止使能 ADC 中断。

2.8.84.2 接口定义

函数接口	void LL_ADC_DisableIT (ADC_TypeDef * ADCx, uint32_t Interrupt)
输入	ADC_TypeDef * ADCx uint32_t Interrupt: { LL_ADC_IT_ADRDY, LL_ADC_IT_EOC, LL_ADC_IT_EOS, LL_ADC_IT_OVR, LL_ADC_IT_AWD1, LL_ADC_IT_EOCAL }
输出	无
返回值	无
资源使用	
说明	

2.8.85 LL_ADC_IsEnabledIT_ADRDY

2.8.85.1 功能介绍

检查是否使能 ADC 就绪中断。

2.8.85.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT_ADRDY (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.86 LL_ADC_IsEnabledIT_EOC

2.8.86.1 功能介绍

检查是否使能转换结束中断。

2.8.86.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT_EOC (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.87 LL_ADC_IsEnabledIT_EOS

2.8.87.1 功能介绍

检查是否使能转换序列结束中断。

2.8.87.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT_EOS (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.88 LL_ADC_IsEnabledIT_OVR

2.8.88.1 功能介绍

检查是否使能溢出中断。

2.8.88.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT_OVR (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无

返回值	{0,1}
资源使用	
说明	

2.8.89 LL_ADC_IsEnabledIT_AWD1

2.8.89.1 功能介绍

检查是否使能模拟看门狗中断。

2.8.89.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT_AWD1 (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.90 LL_ADC_IsEnabledIT_EOCAL

2.8.90.1 功能介绍

检查是否使能校准结束中断。

2.8.90.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT_EOCAL (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.91 LL_ADC_IsEnabledIT

2.8.91.1 功能介绍

检查是否使能 ADC 中断。

2.8.91.2 接口定义

函数接口	uint32_t LL_ADC_IsEnabledIT (ADC_TypeDef * ADCx, uint32_t Interrupt)
输入	ADC_TypeDef * ADCx uint32_t Interrupt: { LL_ADC_IT_ADRDY, LL_ADC_IT_EOC, LL_ADC_IT_EOS, LL_ADC_IT_OVR, LL_ADC_IT_AWD1, LL_ADC_IT_EOCAL }
输出	无
返回值	{0,1}
资源使用	
说明	

2.8.92 LL_ADC_DeInit

2.8.92.1 功能介绍

清除 ADC 初始化参数。

2.8.92.2 接口定义

函数接口	ErrorStatus LL_ADC_DeInit (ADC_TypeDef * ADCx)
输入	ADC_TypeDef * ADCx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.8.93 LL_ADC_Init

2.8.93.1 功能介绍

ADC 初始化。

2.8.93.2 接口定义

函数接口	ErrorStatus LL_ADC_Init (ADC_TypeDef * ADCx, LL_ADC_InitTypeDef * ADC_InitStruct)
输入	ADC_TypeDef * ADCx LL_ADC_InitTypeDef * ADC_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.8.94 LL_ADC_StructInit

2.8.94.1 功能介绍

ADC 结构体初始化。

2.8.94.2 接口定义

函数接口	void LL_ADC_StructInit (LL_ADC_InitTypeDef * ADC_InitStruct)
输入	LL_ADC_InitTypeDef * ADC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.8.95 LL_ADC_REG_Init

2.8.95.1 功能介绍

初始化 ADC 规则组的一些特性。

2.8.95.2 接口定义

函数接口	ErrorStatus LL_ADC_REG_Init (ADC_TypeDef * ADCx, LL_ADC_REG_InitTypeDef * ADC_RegInitStruct)
------	---

输入	ADC_TypeDef * ADCx LL_ADC_REG_InitTypeDef * ADC_RegInitStruct
输出	无
返回值	无
资源使用	
说明	

2.8.96 LL_ADC_REG_StructInit

2.8.96.1 功能介绍

ADC_RegInitStruct 字段设置为默认值。

2.8.96.2 接口定义

函数接口	void LL_ADC_REG_StructInit (LL_ADC_REG_InitTypeDef * ADC_RegInitStruct)
输入	LL_ADC_InitTypeDef * ADC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.9 ATK 模块

属性	类型	字段名	含义
ATK_TypeDef			
读写	uint32_t	CRA	TKA 控制寄存器
读写	uint32_t	CRB	TKB 控制寄存器
读写	uint32_t	ISR	TK 状态寄存器
读写	uint32_t	CHS	TK 启动通道选择寄存器
读写	uint32_t	CHAEN	TKA 通道使能寄存器
读写	uint32_t	CHBEN	TKB 通道使能寄存器
读写	uint32_t	ITRIMA1	TKA 通道调试参数 1
读写	uint32_t	ITRIMA2	TKA 通道调试参数 2
读写	uint32_t	ITRIMA3	TKA 通道调试参数 3
读写	uint32_t	ITRIMA4	TKA 通道调试参数

			4
只读	uint32_t	ITRIMA5	TKA 通道调试参数 5
读写	uint32_t	ITRIMA6	TKA 通道调试参数 6
读写	uint32_t	ITRIMB1	TKB 通道调试参数 1
读写	uint32_t	ITRIMB2	TKB 通道调试参数 2
读写	uint32_t	ITRIMB3	TKB 通道调试参数 3
读写	uint32_t	ITRIMB4	TKB 通道调试参数 4
读写	uint32_t	ITRIMB5	TKB 通道调试参数 5
读写	uint32_t	ITRIMB6	TKB 通道调试参数 6
只读	uint32_t	RAMA	TKA RAM 1~20
只读	uint32_t	RAMB	TKB RAM 1~20

2.9.1 LL_ATK_SetClockFrequency

2.9.1.1 功能介绍

设置触摸键 A、B 时钟频率。

2.9.1.2 接口定义

函数接口	void LL_ATK_SetClockFrequency (ATK_TypeDef * ATKx, uint32_t Port, uint32_t FrqValue)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t FrqValue: { LL_ATK_FREQVALUE_0, LL_ATK_FREQVALUE_1, LL_ATK_FREQVALUE_2, LL_ATK_FREQVALUE_3, LL_ATK_FREQVALUE_4, LL_ATK_FREQVALUE_5, LL_ATK_FREQVALUE_6, LL_ATK_FREQVALUE_7, }
输出	无
返回值	无
资源使用	
说明	

2.9.2 LL_ATK_GetClockFrequency

2.9.2.1 功能介绍

获取触摸键 A、B 时钟频率。

2.9.2.2 接口定义

函数接口	uint32_t LL_ATK_GetClockFrequency (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{ LL_ATK_FREQVALUE_0, LL_ATK_FREQVALUE_1, LL_ATK_FREQVALUE_2, LL_ATK_FREQVALUE_3, LL_ATK_FREQVALUE_4, LL_ATK_FREQVALUE_5, LL_ATK_FREQVALUE_6, LL_ATK_FREQVALUE_7, }
资源使用	
说明	

2.9.3 LL_ATK_EnableClockFrequencyAutoSet

2.9.3.1 功能介绍

使能触键 A 或 B 时钟频率自动设置。。

2.9.3.2 接口定义

函数接口	void LL_ATK_EnableClockFrequencyAutoSet (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.4 LL_ATK_DisableClockFrequencyAutoSet

2.9.4.1 功能介绍

禁止使能触键 A 或 B 时钟频率自动设置。。

2.9.4.2 接口定义

函数接口	void LL_ATK_DisableClockFrequencyAutoSet (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无

资源使用	
说明	

2.9.5 LL_ATK_IsEnabledClockFrequencyAutoSet

2.9.5.1 功能介绍

检查是否使能触键 A 或 B 时钟频率自动设置。。

2.9.5.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledClockFrequencyAutoSet (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.6 LL_ATK_SetScanningLength

2.9.6.1 功能介绍

设置触摸键 A、B 扫描长度。

2.9.6.2 接口定义

函数接口	void LL_ATK_SetScanningLength (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Length)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Length: [0x000, 0xFFFF]
输出	无
返回值	无
资源使用	
说明	

2.9.7 LL_ATK_GetScanningLength

2.9.7.1 功能介绍

获取触摸键 A、B 扫描长度。

2.9.7.2 接口定义

函数接口	uint32_t LL_ATK_GetScanningLength (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无

返回值	[0x000, 0xFFF]
资源使用	
说明	

2.9.8 LL_ATK_EnableExternalCap

2.9.8.1 功能介绍

使能触摸键 A、B 外部电容。

2.9.8.2 接口定义

函数接口	void LL_ATK_EnableExternalCap (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.9 LL_ATK_DisableExternalCap

2.9.9.1 功能介绍

禁止使能触摸键 A、B 外部电容。

2.9.9.2 接口定义

函数接口	void LL_ATK_DisableExternalCap (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.10 LL_ATK_IsEnabledExternalCap

2.9.10.1 功能介绍

检查是否使能触摸键 A、B 外部电容。

2.9.10.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledExternalCap (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}

资源使用	
说明	

2.9.11 LL_ATK_EnableAutoRerun

2.9.11.1 功能介绍

使能触摸键 A、B 自动重新启动。

2.9.11.2 接口定义

函数接口	void LL_ATK_EnableAutoRerun (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.12 LL_ATK_DisableAutoRerun

2.9.12.1 功能介绍

禁止使能触摸键 A、B 自动重新启动。

2.9.12.2 接口定义

函数接口	void LL_ATK_DisableAutoRerun (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.13 LL_ATK_IsEnabledAutoRerun

2.9.13.1 功能介绍

检查是否使能触摸键 A、B 自动重新启动。

2.9.13.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledAutoRerun (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.14 LL_ATK_SetScanningMode

2.9.14.1 功能介绍

设置触摸键 A、B 扫描模式。

2.9.14.2 接口定义

函数接口	void LL_ATK_SetScanningMode (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Mode)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Mode: { LL_ATK_MODE_1_22, LL_ATK_MODE_2_20, LL_ATK_MODE_4_10, LL_ATK_MODE_8_5 }
输出	无
返回值	无
资源使用	
说明	

2.9.15 LL_ATK_GetScanningMode

2.9.15.1 功能介绍

获取触摸键 A、B 扫描模式。

2.9.15.2 接口定义

函数接口	uint32_t LL_ATK_GetScanningMode (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{ LL_ATK_MODE_1_22, LL_ATK_MODE_2_20, LL_ATK_MODE_4_10, LL_ATK_MODE_8_5 }
资源使用	
说明	

2.9.16 LL_ATK_SetVoltageReference

2.9.16.1 功能介绍

设置触摸键 A、B 参考电压。

2.9.16.2 接口定义

函数接口	void LL_ATK_SetVoltageReference (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Voltage)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Voltage: { LL_ATK_VOL_2_8, LL_ATK_VOL_3_6 }

输出	无
返回值	无
资源使用	
说明	

2.9.17 LL_ATK_GetVoltageReference

2.9.17.1 功能介绍

获取触摸键 A、B 参考电压。

2.9.17.2 接口定义

函数接口	uint32_t LL_ATK_GetVoltageReference (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{ LL_ATK_VOL_2_8, LL_ATK_VOL_3_6 }
资源使用	
说明	

2.9.18 LL_ATK_SetChannelTrimming

2.9.18.1 功能介绍

设置触摸键 A、B 通道调试值。

2.9.18.2 接口定义

函数接口	void LL_ATK_SetChannelTrimming (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Channelx, uint32_t Data)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Channelx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP } uint32_t Data: [0x00, 0x7F]
输出	无

返回值	无
资源使用	
说明	

2.9.19 LL_ATK_GetChannelTrimming

2.9.19.1 功能介绍

获取触摸键 A、B 通道调试值。

2.9.19.2 接口定义

函数接口	uint32_t LL_ATK_GetChannelTrimming (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Channelx)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Channelx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP }
输出	无
返回值	[0x00, 0x7F]
资源使用	
说明	

2.9.20 LL_ATK_EnableClockfrequencySpread

2.9.20.1 功能介绍

使能触摸键 A、B 时钟频率扩展。

2.9.20.2 接口定义

函数接口	void LL_ATK_EnableClockfrequencySpread (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.21 LL_ATK_DisableClockfrequencySpread

2.9.21.1 功能介绍

禁止使能触摸键 A、B 时钟频率扩展。

2.9.21.2 接口定义

函数接口	void LL_ATK_DisableClockfrequencySpread (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.22 LL_ATK_IsEnabledClockfrequencySpread

2.9.22.1 功能介绍

检查是否使能触摸键 A、B 时钟频率扩展。

2.9.22.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledClockfrequencySpread (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.23 LL_ATK_EnableIT_Finished

2.9.23.1 功能介绍

使能触摸键 A、B 转换结束中断。

2.9.23.2 接口定义

函数接口	void LL_ATK_EnableIT_Finished (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.24 LL_ATK_DisableIT_Finished

2.9.24.1 功能介绍

禁止使能触摸键 A、B 转换结束中断。

2.9.24.2 接口定义

函数接口	void LL_ATK_DisableIT_Finished (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.25 LL_ATK_IsEnabledIT_Finished

2.9.25.1 功能介绍

检查是否使能触摸键 A、B 转换结束中断。

2.9.25.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledIT_Finished (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.26 LL_ATK_IsActiveFlag_EOC

2.9.26.1 功能介绍

检查触摸键 A、B 是否转换结束。

2.9.26.2 接口定义

函数接口	uint32_t LL_ATK_IsActiveFlag_EOC (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.27 LL_ATK_IsActiveFlag_IT

2.9.27.1 功能介绍

检查触摸键 A、B 是否产生中断标志。

2.9.27.2 接口定义

函数接口	uint32_t LL_ATK_IsActiveFlag_IT (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.28 LL_ATK_ClearFlag_IT

2.9.28.1 功能介绍

清除触摸键 A、B 中断标志。

2.9.28.2 接口定义

函数接口	void LL_ATK_ClearFlag_IT (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.29 LL_ATK_EnableChannel

2.9.29.1 功能介绍

使能触摸键 A、B 通道。

2.9.29.2 接口定义

函数接口	void LL_ATK_EnableChannel (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Channelx)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Channelx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9,

	LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP }
输出	无
返回值	无
资源使用	
说明	

2.9.30 LL_ATK_DisableChannel

2.9.30.1 功能介绍

禁止使能触摸键 A、B 通道。

2.9.30.2 接口定义

函数接口	void LL_ATK_DisableChannel (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Channelx)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Channelx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP }
输出	无
返回值	无
资源使用	
说明	

2.9.31 LL_ATK_IsEnabledChannel

2.9.31.1 功能介绍

检查是否使能触摸键 A、B 通道。

2.9.31.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledChannel (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Channelx)
------	---

输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Channelx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.32 LL_ATK_Set1stChannel

2.9.32.1 功能介绍

设置触摸键 A、B 第一个扫描通道。

2.9.32.2 接口定义

函数接口	void LL_ATK_Set1stChannel (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Channelx)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Channelx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP }
输出	无
返回值	无
资源使用	

说明	
----	--

2.9.33 LL_ATK_Get1stChannel

2.9.33.1 功能介绍

获取触摸键 A、B 第一个扫描通道。

2.9.33.2 接口定义

函数接口	uint32_t LL_ATK_Get1stChannel (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{ LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20, LL_ATK_CHANNEL_CAP }
资源使用	
说明	

2.9.34 LL_ATK_EnableABPinExchange

2.9.34.1 功能介绍

使能触摸键 A、B 引脚互换。

2.9.34.2 接口定义

函数接口	void LL_ATK_EnableABPinExchange (ATK_TypeDef * ATKx, uint32_t Pinx)
输入	ATK_TypeDef * ATKx uint32_t Pinx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20 }

输出	无
返回值	无
资源使用	
说明	

2.9.35 LL_ATK_DisableABPinExchange

2.9.35.1 功能介绍

禁止使能触摸键 A、B 引脚互换。

2.9.35.2 接口定义

函数接口	void LL_ATK_DisableABPinExchange (ATK_TypeDef * ATKx, uint32_t Pinx)
输入	ATK_TypeDef * ATKx uint32_t Pinx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13, LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20 }
输出	无
返回值	无
资源使用	
说明	

2.9.36 LL_ATK_IsEnabledABPinExchange

2.9.36.1 功能介绍

检查是否使能触摸键 A、B 引脚互换。

2.9.36.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledABPinExchange (ATK_TypeDef * ATKx, uint32_t Pinx)
输入	ATK_TypeDef * ATKx uint32_t Pinx: { LL_ATK_CHANNEL_0, LL_ATK_CHANNEL_1, LL_ATK_CHANNEL_2, LL_ATK_CHANNEL_3, LL_ATK_CHANNEL_4, LL_ATK_CHANNEL_5, LL_ATK_CHANNEL_6, LL_ATK_CHANNEL_7, LL_ATK_CHANNEL_8, LL_ATK_CHANNEL_9, LL_ATK_CHANNEL_10, LL_ATK_CHANNEL_11, LL_ATK_CHANNEL_12, LL_ATK_CHANNEL_13,

	LL_ATK_CHANNEL_14, LL_ATK_CHANNEL_15, LL_ATK_CHANNEL_16, LL_ATK_CHANNEL_17, LL_ATK_CHANNEL_18, LL_ATK_CHANNEL_19, LL_ATK_CHANNEL_20 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.37 LL_ATK_Enable

2.9.37.1 功能介绍

使能触摸键 A、B。

2.9.37.2 接口定义

函数接口	void LL_ATK_Enable (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.38 LL_ATK_Disable

2.9.38.1 功能介绍

禁止使能触摸键 A、B。

2.9.38.2 接口定义

函数接口	void LL_ATK_Disable (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.39 LL_ATK_IsEnabled

2.9.39.1 功能介绍

检查是否使能触摸键 A、B。

2.9.39.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabled (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port:

	{ LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.40 LL_ATK_EnableDMA

2.9.40.1 功能介绍

触摸键 A、B 使能 DMA。

2.9.40.2 接口定义

函数接口	void LL_ATK_EnableDMA (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.41 LL_ATK_DisableDMA

2.9.41.1 功能介绍

禁止触摸键 A、B 使能 DMA。

2.9.41.2 接口定义

函数接口	void LL_ATK_DisableDMA (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.42 LL_ATK_IsEnabledDMA

2.9.42.1 功能介绍

检查是否触摸键 A、B 使能 DMA。

2.9.42.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledDMA (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}

资源使用	
说明	

2.9.43 LL_ATK_GetRAMValue

2.9.43.1 功能介绍

火气触摸键 A、B RAM 值。

2.9.43.2 接口定义

函数接口	uint32_t LL_ATK_GetRAMValue (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Index)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Index: [0, 39]
输出	无
返回值	[0x0000, 0xFFFF]
资源使用	
说明	

2.9.44 LL_ATK_GetRegAddr

2.9.44.1 功能介绍

火气触摸键 A、B RAM 地址。

2.9.44.2 接口定义

函数接口	uint32_t LL_ATK_GetRegAddr (ATK_TypeDef * ATKx, uint32_t Port, uint32_t Index)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B } uint32_t Index: [0, 39]
输出	无
返回值	寄存器地址
资源使用	
说明	

2.9.45 LL_ATK_EnableShield

2.9.45.1 功能介绍

使能触摸键 A、B 屏蔽。

2.9.45.2 接口定义

函数接口	void LL_ATK_EnableShild (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }

输出	无
返回值	无
资源使用	
说明	

2.9.46 LL_ATK_DisableShield

2.9.46.1 功能介绍

禁止使能触摸键 A、B 屏蔽。

2.9.46.2 接口定义

函数接口	void LL_ATK_DisableShield (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.47 LL_ATK_IsEnabledShield

2.9.47.1 功能介绍

检查是否使能触摸键 A、B 屏蔽。

2.9.47.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledShield (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.9.48 LL_ATK_EnableConvert

2.9.48.1 功能介绍

使能触摸键 A、B 转换。

2.9.48.2 接口定义

函数接口	void LL_ATK_EnableConvert (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	

说明	
----	--

2.9.49 LL_ATK_DisableConvert

2.9.49.1 功能介绍

禁止使能触摸键 A、B 转换。

2.9.49.2 接口定义

函数接口	void LL_ATK_DisableConvert (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	无
资源使用	
说明	

2.9.50 LL_ATK_IsEnabledConvert

2.9.50.1 功能介绍

检查是否使能触摸键 A、B 转换。

2.9.50.2 接口定义

函数接口	uint32_t LL_ATK_IsEnabledConvert (ATK_TypeDef * ATKx, uint32_t Port)
输入	ATK_TypeDef * ATKx uint32_t Port: { LL_ATK_PORT_A, LL_ATK_PORT_B }
输出	无
返回值	{0,1}
资源使用	
说明	

2.10 BUS 模块

2.10.1 LL_AHB1_GRP1_EnableClock

2.10.1.1 功能介绍

使能 AHB 外设时钟。

2.10.1.2 接口定义

函数接口	void LL_AHB1_GRP1_EnableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_AHB1_GRP1_PERIPH_DMA1, LL_AHB1_GRP1_PERIPH_CRC, LL_AHB1_GRP1_PERIPH_PLA, LL_AHB1_GRP1_PERIPH_HDIV }
输出	无
返回值	无
资源使用	

说明	
----	--

2.10.2 LL_AHB1_GRP1_IsEnabledClock

2.10.2.1 功能介绍

检查是否使能 AHB 外设时钟。

2.10.2.2 接口定义

函数接口	uint32_t LL_AHB1_GRP1_IsEnabledClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_AHB1_GRP1_PERIPH_DMA1, LL_AHB1_GRP1_PERIPH_CRC, LL_AHB1_GRP1_PERIPH_PLA, LL_AHB1_GRP1_PERIPH_HDIV }
输出	无
返回值	{0,1}
资源使用	
说明	

2.10.3 LL_AHB1_GRP1_DisableClock

2.10.3.1 功能介绍

禁止使能 AHB 外设时钟。

2.10.3.2 接口定义

函数接口	void LL_AHB1_GRP1_DisableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_AHB1_GRP1_PERIPH_DMA1, LL_AHB1_GRP1_PERIPH_CRC, LL_AHB1_GRP1_PERIPH_PLA, LL_AHB1_GRP1_PERIPH_HDIV }
输出	无
返回值	无
资源使用	
说明	

2.10.4 LL_AHB1_GRP1_ForceReset

2.10.4.1 功能介绍

强制 AHB 外设复位。

2.10.4.2 接口定义

函数接口	void LL_AHB1_GRP1_ForceReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_AHB1_GRP1_PERIPH_DMA1, LL_AHB1_GRP1_PERIPH_CRC, LL_AHB1_GRP1_PERIPH_PLA, LL_AHB1_GRP1_PERIPH_HDIV }
输出	无
返回值	无
资源使用	
说明	

2.10.5 LL_AHB1_GRP1_ReleaseReset

2.10.5.1 功能介绍

释放 AHB 外设复位。

2.10.5.2 接口定义

函数接口	void LL_AHB1_GRP1_ReleaseReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_AHB1_GRP1_PERIPH_DMA1, LL_AHB1_GRP1_PERIPH_CRC, LL_AHB1_GRP1_PERIPH_PLA, LL_AHB1_GRP1_PERIPH_HDIV }
输出	无
返回值	无
资源使用	
说明	

2.10.6 LL_APB1_GRP1_EnableClock

2.10.6.1 功能介绍

使能 APB1 外设时钟。

2.10.6.2 接口定义

函数接口	void LL_APB1_GRP1_EnableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB1_GRP1_PERIPH_TIM2, LL_APB1_GRP1_PERIPH_TIM3, LL_APB1_GRP1_PERIPH_TIM4, LL_APB1_GRP1_PERIPH_TIM5, LL_APB1_GRP1_PERIPH_TIM6, LL_APB1_GRP1_PERIPH_TIM7, LL_APB1_GRP1_PERIPH_LCD, LL_APB1_GRP1_PERIPH_RTC, LL_APB1_GRP1_PERIPH_WWDG, LL_APB1_GRP1_PERIPH_SPI2, LL_APB1_GRP1_PERIPH_ATK, LL_APB1_GRP1_PERIPH_USART2, LL_APB1_GRP1_PERIPH_USART3, LL_APB1_GRP1_PERIPH_I2C1, LL_APB1_GRP1_PERIPH_I2C2, LL_APB1_GRP1_PERIPH_LPUART1, LL_APB1_GRP1_PERIPH_PWR, LL_APB1_GRP1_PERIPH_SAC, LL_APB1_GRP1_PERIPH_LPTIM1 }
输出	无
返回值	无
资源使用	
说明	

2.10.7 LL_APB1_GRP1_IsEnabledClock

2.10.7.1 功能介绍

检查是否使能 APB1 外设时钟。

2.10.7.2 接口定义

函数接口	uint32_t LL_APB1_GRP1_IsEnabledClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB1_GRP1_PERIPH_TIM2, LL_APB1_GRP1_PERIPH_TIM3, LL_APB1_GRP1_PERIPH_TIM4, LL_APB1_GRP1_PERIPH_TIM5,

	LL_APB1_GRP1_PERIPH_TIM6, LL_APB1_GRP1_PERIPH_TIM7, LL_APB1_GRP1_PERIPH_LCD, LL_APB1_GRP1_PERIPH_RTC, LL_APB1_GRP1_PERIPH_WWDG, LL_APB1_GRP1_PERIPH_SPI2, LL_APB1_GRP1_PERIPH_ATK, LL_APB1_GRP1_PERIPH_USART2, LL_APB1_GRP1_PERIPH_USART3, LL_APB1_GRP1_PERIPH_I2C1, LL_APB1_GRP1_PERIPH_I2C2, LL_APB1_GRP1_PERIPH_LPUART1, LL_APB1_GRP1_PERIPH_PWR, LL_APB1_GRP1_PERIPH_SAC, LL_APB1_GRP1_PERIPH_LPTIM1 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.10.8 LL_APB1_GRP1_DisableClock

2.10.8.1 功能介绍

禁止使能 APB1 外设时钟。

2.10.8.2 接口定义

函数接口	void LL_APB1_GRP1_DisableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB1_GRP1_PERIPH_TIM2, LL_APB1_GRP1_PERIPH_TIM3, LL_APB1_GRP1_PERIPH_TIM4, LL_APB1_GRP1_PERIPH_TIM5, LL_APB1_GRP1_PERIPH_TIM6, LL_APB1_GRP1_PERIPH_TIM7, LL_APB1_GRP1_PERIPH_LCD, LL_APB1_GRP1_PERIPH_RTC, LL_APB1_GRP1_PERIPH_WWDG, LL_APB1_GRP1_PERIPH_SPI2, LL_APB1_GRP1_PERIPH_ATK, LL_APB1_GRP1_PERIPH_USART2, LL_APB1_GRP1_PERIPH_USART3, LL_APB1_GRP1_PERIPH_I2C1, LL_APB1_GRP1_PERIPH_I2C2, LL_APB1_GRP1_PERIPH_LPUART1, LL_APB1_GRP1_PERIPH_PWR, LL_APB1_GRP1_PERIPH_SAC, LL_APB1_GRP1_PERIPH_LPTIM1 }
输出	无
返回值	无
资源使用	
说明	

2.10.9 LL_APB1_GRP1_ForceReset

2.10.9.1 功能介绍

强制 APB1 外设复位。

2.10.9.2 接口定义

函数接口	void LL_APB1_GRP1_ForceReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB1_GRP1_PERIPH_TIM2, LL_APB1_GRP1_PERIPH_TIM3, LL_APB1_GRP1_PERIPH_TIM4, LL_APB1_GRP1_PERIPH_TIM5, LL_APB1_GRP1_PERIPH_TIM6, LL_APB1_GRP1_PERIPH_TIM7,

	LL_APB1_GRP1_PERIPH_LCD, LL_APB1_GRP1_PERIPH_RTC, LL_APB1_GRP1_PERIPH_WWDG, LL_APB1_GRP1_PERIPH_SPI2, LL_APB1_GRP1_PERIPH_ATK, LL_APB1_GRP1_PERIPH_USART2, LL_APB1_GRP1_PERIPH_USART3, LL_APB1_GRP1_PERIPH_I2C1, LL_APB1_GRP1_PERIPH_I2C2, LL_APB1_GRP1_PERIPH_LPUART1, LL_APB1_GRP1_PERIPH_PWR, LL_APB1_GRP1_PERIPH_SAC, LL_APB1_GRP1_PERIPH_LPTIM1 }
输出	无
返回值	无
资源使用	
说明	

2.10.10 LL_APB1_GRP1_ReleaseReset

2.10.10.1 功能介绍

释放 APB1 外设复位。

2.10.10.2 接口定义

函数接口	void LL_APB1_GRP1_ReleaseReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB1_GRP1_PERIPH_TIM2, LL_APB1_GRP1_PERIPH_TIM3, LL_APB1_GRP1_PERIPH_TIM4, LL_APB1_GRP1_PERIPH_TIM5, LL_APB1_GRP1_PERIPH_TIM6, LL_APB1_GRP1_PERIPH_TIM7, LL_APB1_GRP1_PERIPH_LCD, LL_APB1_GRP1_PERIPH_RTC, LL_APB1_GRP1_PERIPH_WWDG, LL_APB1_GRP1_PERIPH_SPI2, LL_APB1_GRP1_PERIPH_ATK, LL_APB1_GRP1_PERIPH_USART2, LL_APB1_GRP1_PERIPH_USART3, LL_APB1_GRP1_PERIPH_I2C1, LL_APB1_GRP1_PERIPH_I2C2, LL_APB1_GRP1_PERIPH_LPUART1, LL_APB1_GRP1_PERIPH_PWR, LL_APB1_GRP1_PERIPH_SAC, LL_APB1_GRP1_PERIPH_LPTIM1 }
输出	无
返回值	无
资源使用	
说明	

2.10.11 LL_APB2_GRP1_EnableClock

2.10.11.1 功能介绍

使能 APB2 外设时钟。

2.10.11.2 接口定义

函数接口	void LL_APB2_GRP1_EnableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB2_GRP1_PERIPH_SYS, LL_APB2_GRP1_PERIPH_TIM1, LL_APB2_GRP1_PERIPH_SPI1, LL_APB2_GRP1_PERIPH_USART1, LL_APB2_GRP1_PERIPH_ADC, LL_APB2_GRP1_PERIPH_DBG }
输出	无

返回值	无
资源使用	
说明	

2.10.12 LL_APB2_GRP1_IsEnabledClock

2.10.12.1 功能介绍

检查是否使能 APB2 外设时钟。

2.10.12.2 接口定义

函数接口	uint32_t LL_APB2_GRP1_IsEnabledClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB2_GRP1_PERIPH_SYS, LL_APB2_GRP1_PERIPH_TIM1, LL_APB2_GRP1_PERIPH_SPI1, LL_APB2_GRP1_PERIPH_USART1, LL_APB2_GRP1_PERIPH_ADC, LL_APB2_GRP1_PERIPH_DBG }
输出	无
返回值	{0,1}
资源使用	
说明	

2.10.13 LL_APB2_GRP1_DisableClock

2.10.13.1 功能介绍

禁止使能 APB2 外设时钟。

2.10.13.2 接口定义

函数接口	void LL_APB2_GRP1_DisableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB2_GRP1_PERIPH_SYS, LL_APB2_GRP1_PERIPH_TIM1, LL_APB2_GRP1_PERIPH_SPI1, LL_APB2_GRP1_PERIPH_USART1, LL_APB2_GRP1_PERIPH_ADC, LL_APB2_GRP1_PERIPH_DBG }
输出	无
返回值	无
资源使用	
说明	

2.10.14 LL_APB2_GRP1_ForceReset

2.10.14.1 功能介绍

强制 APB2 外设复位。

2.10.14.2 接口定义

函数接口	void LL_APB2_GRP1_ForceReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB2_GRP1_PERIPH_SYS, LL_APB2_GRP1_PERIPH_TIM1, LL_APB2_GRP1_PERIPH_SPI1, LL_APB2_GRP1_PERIPH_USART1, LL_APB2_GRP1_PERIPH_ADC, LL_APB2_GRP1_PERIPH_DBG }
输出	无

返回值	无
资源使用	
说明	

2.10.15 LL_APB2_GRP1_ReleaseReset

2.10.15.1 功能介绍

释放 APB2 外设复位。

2.10.15.2 接口定义

函数接口	void LL_APB2_GRP1_ReleaseReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_APB2_GRP1_PERIPH_SYS, LL_APB2_GRP1_PERIPH_TIM1, LL_APB2_GRP1_PERIPH_SPI1, LL_APB2_GRP1_PERIPH_USART1, LL_APB2_GRP1_PERIPH_ADC, LL_APB2_GRP1_PERIPH_DBG }
输出	无
返回值	无
资源使用	
说明	

2.10.16 LL_IOP_GRP1_EnableClock

2.10.16.1 功能介绍

使能 IO 外设时钟。

2.10.16.2 接口定义

函数接口	void LL_IOP_GRP1_EnableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_IOP_GRP1_PERIPH_GPIOA, LL_IOP_GRP1_PERIPH_GPIOB, LL_IOP_GRP1_PERIPH_GPIOC, LL_IOP_GRP1_PERIPH_GPIOD }
输出	无
返回值	无
资源使用	
说明	

2.10.17 LL_IOP_GRP1_IsEnabledClock

2.10.17.1 功能介绍

检查是否使能 IO 外设时钟。

2.10.17.2 接口定义

函数接口	uint32_t LL_IOP_GRP1_IsEnabledClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_IOP_GRP1_PERIPH_GPIOA, LL_IOP_GRP1_PERIPH_GPIOB, LL_IOP_GRP1_PERIPH_GPIOC, LL_IOP_GRP1_PERIPH_GPIOD }
输出	无
返回值	{0,1}
资源使用	

说明	
----	--

2.10.18 LL_IOP_GRP1_DisableClock

2.10.18.1 功能介绍

禁止使能 IO 外设时钟。

2.10.18.2 接口定义

函数接口	void LL_APB2_GRP1_DisableClock (uint32_t Periphs)
输入	uint32_t Periphs : { LL_IOP_GRP1_PERIPH_GPIOA, LL_IOP_GRP1_PERIPH_GPIOB, LL_IOP_GRP1_PERIPH_GPIOC, LL_IOP_GRP1_PERIPH_GPIOD }
输出	无
返回值	无
资源使用	
说明	

2.10.19 LL_IOP_GRP1_ForceReset

2.10.19.1 功能介绍

强制 IO 外设复位。

2.10.19.2 接口定义

函数接口	void LL_IOP_GRP1_ForceReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_IOP_GRP1_PERIPH_GPIOA, LL_IOP_GRP1_PERIPH_GPIOB, LL_IOP_GRP1_PERIPH_GPIOC, LL_IOP_GRP1_PERIPH_GPIOD }
输出	无
返回值	无
资源使用	
说明	

2.10.20 LL_IOP_GRP1_ReleaseReset

2.10.20.1 功能介绍

释放 IO 外设复位。

2.10.20.2 接口定义

函数接口	void LL_IOP_GRP1_ReleaseReset (uint32_t Periphs)
输入	uint32_t Periphs : { LL_IOP_GRP1_PERIPH_GPIOA, LL_IOP_GRP1_PERIPH_GPIOB, LL_IOP_GRP1_PERIPH_GPIOC, LL_IOP_GRP1_PERIPH_GPIOD }
输出	无
返回值	无
资源使用	
说明	

2.11 COMP 模块

属性	类型	字段名	含义
COMP_TypeDef			
读写	uint32_t	CSR	控制 和 状态 寄存器
LL_COMP_InitTypeDef			
读写	uint32_t	WindowMode	窗口比较器模式
读写	uint32_t	WindowOutput	窗口模式输出
读写	uint32_t	InputPlus	正相输入信号
读写	uint32_t	InputMinus	反相输入信号
读写	uint32_t	OutputPolarity	输出极性
读写	uint32_t	OutputFilterTimes	输出滤波时间
读写	FunctionalState	OutputFilterEnable	输出滤波使能

2.11.1 LL_COMP_ConfigInputs

2.11.1.1 功能介绍

设置比较器正相和反相输入。

2.11.1.2 接口定义

函数接口	void LL_COMP_ConfigInputs (COMP_TypeDef * COMPx, uint32_t InputMinus, uint32_t InputPlus)
输入	COMP_TypeDef * COMPx uint32_t InputMinus: { LL_COMP_INPUT_MINUS_IO1, LL_COMP_INPUT_MINUS_IO2, LL_COMP_INPUT_MINUS_IO3, LL_COMP_INPUT_MINUS_IO4, LL_COMP_INPUT_MINUS_IO5, LL_COMP_INPUT_MINUS_IO6, LL_COMP_INPUT_MINUS_IO7, LL_COMP_INPUT_MINUS_IO8, LL_COMP_INPUT_MINUS_IO9, LL_COMP_INPUT_MINUS_IO10, LL_COMP_INPUT_MINUS_DAC } uint32_t InputPlus: { LL_COMP_INPUT_PLUS_IO1, LL_COMP_INPUT_PLUS_IO2, LL_COMP_INPUT_PLUS_IO3, LL_COMP_INPUT_PLUS_IO4, LL_COMP_INPUT_PLUS_IO5, LL_COMP_INPUT_PLUS_IO6, LL_COMP_INPUT_PLUS_IO7, LL_COMP_INPUT_PLUS_IO8, LL_COMP_INPUT_PLUS_IO9, LL_COMP_INPUT_PLUS_IO10, LL_COMP_INPUT_PLUS_OPA }
输出	无
返回值	无
资源使用	
说明	

2.11.2 LL_COMP_SetInputPlus

2.11.2.1 功能介绍

设置比较器正相输入。

2.11.2.2 接口定义

函数接口	void LL_COMP_SetInputPlus (COMP_TypeDef * COMPx, uint32_t InputPlus)
输入	COMP_TypeDef * COMPx uint32_t InputPlus: { LL_COMP_INPUT_PLUS_IO1, LL_COMP_INPUT_PLUS_IO2, LL_COMP_INPUT_PLUS_IO3, LL_COMP_INPUT_PLUS_IO4, LL_COMP_INPUT_PLUS_IO5, LL_COMP_INPUT_PLUS_IO6, LL_COMP_INPUT_PLUS_IO7, LL_COMP_INPUT_PLUS_IO8, LL_COMP_INPUT_PLUS_IO9, LL_COMP_INPUT_PLUS_IO10, LL_COMP_INPUT_PLUS_OPA }
输出	无
返回值	无
资源使用	
说明	

2.11.3 LL_COMP_GetInputPlus

2.11.3.1 功能介绍

获取比较器正相输入。

2.11.3.2 接口定义

函数接口	uint32_t LL_COMP_GetInputPlus (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{ LL_COMP_INPUT_PLUS_IO1, LL_COMP_INPUT_PLUS_IO2, LL_COMP_INPUT_PLUS_IO3, LL_COMP_INPUT_PLUS_IO4, LL_COMP_INPUT_PLUS_IO5, LL_COMP_INPUT_PLUS_IO6, LL_COMP_INPUT_PLUS_IO7, LL_COMP_INPUT_PLUS_IO8, LL_COMP_INPUT_PLUS_IO9, LL_COMP_INPUT_PLUS_IO10, LL_COMP_INPUT_PLUS_OPA }
资源使用	
说明	

2.11.4 LL_COMP_SetInputMinus

2.11.4.1 功能介绍

设置比较器反相输入。

2.11.4.2 接口定义

函数接口	void LL_COMP_SetInputMinus (COMP_TypeDef * COMPx, uint32_t InputMinus)
输入	COMP_TypeDef * COMPx

	uint32_t InputMinus: { LL_COMP_INPUT_MINUS_IO1, LL_COMP_INPUT_MINUS_IO2, LL_COMP_INPUT_MINUS_IO3, LL_COMP_INPUT_MINUS_IO4, LL_COMP_INPUT_MINUS_IO5, LL_COMP_INPUT_MINUS_IO6, LL_COMP_INPUT_MINUS_IO7, LL_COMP_INPUT_MINUS_IO8, LL_COMP_INPUT_MINUS_IO9, LL_COMP_INPUT_MINUS_IO10, LL_COMP_INPUT_MINUS_DAC }
输出	无
返回值	无
资源使用	
说明	

2.11.5 LL_COMP_GetInputMinus

2.11.5.1 功能介绍

获取比较器反相输入。

2.11.5.2 接口定义

函数接口	uint32_t LL_COMP_GetInputMinus (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{ LL_COMP_INPUT_MINUS_IO1, LL_COMP_INPUT_MINUS_IO2, LL_COMP_INPUT_MINUS_IO3, LL_COMP_INPUT_MINUS_IO4, LL_COMP_INPUT_MINUS_IO5, LL_COMP_INPUT_MINUS_IO6, LL_COMP_INPUT_MINUS_IO7, LL_COMP_INPUT_MINUS_IO8, LL_COMP_INPUT_MINUS_IO9, LL_COMP_INPUT_MINUS_IO10, LL_COMP_INPUT_MINUS_DAC }
资源使用	
说明	

2.11.6 LL_COMP_SetWindowMode

2.11.6.1 功能介绍

设置比较器窗口比较模式。

2.11.6.2 接口定义

函数接口	void LL_COMP_SetWindowMode (COMP_TypeDef * COMPx, uint32_t WindowMode)
输入	COMP_TypeDef * COMPx uint32_t WindowMode: { LL_COMP_WINDOWMODE_OWN, LL_COMP_WINDOWMODE_OTHER }
输出	无
返回值	无
资源使用	
说明	

2.11.7 LL_COMP_GetWindowMode

2.11.7.1 功能介绍

获取比较器窗口比较模式。

2.11.7.2 接口定义

函数接口	uint32_t LL_COMP_GetWindowMode (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{ LL_COMP_WINDOWMODE_OWN, LL_COMP_WINDOWMODE_OTHER }
资源使用	
说明	

2.11.8 LL_COMP_SetOutputPolarity

2.11.8.1 功能介绍

设置比较器输出极性。

2.11.8.2 接口定义

函数接口	void LL_COMP_SetOutputPolarity (COMP_TypeDef * COMPx, uint32_t OutputPolarity)
输入	COMP_TypeDef * COMPx uint32_t OutputPolarity: { LL_COMP_OUTPUTPOL_NONINVERTED, LL_COMP_OUTPUTPOL_INVERTED }
输出	无
返回值	无
资源使用	
说明	

2.11.9 LL_COMP_GetOutputPolarity

2.11.9.1 功能介绍

获取比较器输出极性。

2.11.9.2 接口定义

函数接口	uint32_t LL_COMP_GetOutputPolarity (COMP_TypeDef * COMPx, uint32_t OutputPolarity)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{ LL_COMP_OUTPUTPOL_NONINVERTED, LL_COMP_OUTPUTPOL_INVERTED }
资源使用	
说明	

2.11.10 LL_COMP_SetWindowOutput

2.11.10.1 功能介绍

设置比较器输出信号选择。

2.11.10.2 接口定义

函数接口	void LL_COMP_SetWindowOutput (COMP_TypeDef * COMPx, uint32_t OutputMode)
输入	COMP_TypeDef * COMPx uint32_t OutputMode: { LL_COMP_WINDOWOUT_SINGLE, LL_COMP_WINDOWOUT_COMBIAN }
输出	无
返回值	无
资源使用	
说明	

2.11.11 LL_COMP_GetWindowOutput

2.11.11.1 功能介绍

获取比较器输出信号选择。

2.11.11.2 接口定义

函数接口	uint32_t LL_COMP_GetWindowOutput (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{ LL_COMP_WINDOWOUT_SINGLE, LL_COMP_WINDOWOUT_COMBIAN }
资源使用	
说明	

2.11.12 LL_COMP_FILTER_Enable

2.11.12.1 功能介绍

使能比较器滤波。

2.11.12.2 接口定义

函数接口	void LL_COMP_FILTER_Enable (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	无
资源使用	
说明	

2.11.13 LL_COMP_FILTER_Disable

2.11.13.1 功能介绍

禁止使能比较器滤波。

2.11.13.2 接口定义

函数接口	void LL_COMP_FILTER_Disable (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	无
资源使用	
说明	

2.11.14 LL_COMP_FILTER_IsEnabled

2.11.14.1 功能介绍

检查是否使能比较器滤波。

2.11.14.2 接口定义

函数接口	uint32_t LL_COMP_FILTER_IsEnabled (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{0,1}
资源使用	
说明	

2.11.15 LL_COMP_SetOutputFilterTimes

2.11.15.1 功能介绍

设置比较器输出滤波时钟周期。

2.11.15.2 接口定义

函数接口	void LL_COMP_SetOutputFilterTimes (COMP_TypeDef * COMPx, uint32_t OutputPolarity)
输入	COMP_TypeDef * COMPx uint32_t OutputPolarity: { LL_COMP_OUTPUTFILTER_CYCLE2, LL_COMP_OUTPUTFILTER_CYCLE4, LL_COMP_OUTPUTFILTER_CYCLE8, LL_COMP_OUTPUTFILTER_CYCLE16, LL_COMP_OUTPUTFILTER_CYCLE32, LL_COMP_OUTPUTFILTER_CYCLE64, LL_COMP_OUTPUTFILTER_CYCLE128, LL_COMP_OUTPUTFILTER_CYCLE256 }
输出	无
返回值	无
资源使用	
说明	

2.11.16 LL_COMP_GetOutputFilterTimes

2.11.16.1 功能介绍

获取比较器输出滤波时钟周期。

2.11.16.2 接口定义

函数接口	uint32_t LL_COMP_GetOutputFilterTimes (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{ LL_COMP_OUTPUTFILTER_CYCLE2, LL_COMP_OUTPUTFILTER_CYCLE4, LL_COMP_OUTPUTFILTER_CYCLE8, LL_COMP_OUTPUTFILTER_CYCLE16, LL_COMP_OUTPUTFILTER_CYCLE32, LL_COMP_OUTPUTFILTER_CYCLE64, LL_COMP_OUTPUTFILTER_CYCLE128, LL_COMP_OUTPUTFILTER_CYCLE256 }
资源使用	
说明	

2.11.17 LL_COMP_Enable

2.11.17.1 功能介绍

使能比较器。

2.11.17.2 接口定义

函数接口	void LL_COMP_Enable (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	无
资源使用	
说明	

2.11.18 LL_COMP_Disable

2.11.18.1 功能介绍

禁止使能比较器。

2.11.18.2 接口定义

函数接口	void LL_COMP_Disable (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	无
资源使用	
说明	

2.11.19 LL_COMP_IsEnabled

2.11.19.1 功能介绍

检查是否使能比较器。

2.11.19.2 接口定义

函数接口	uint32_t LL_COMP_IsEnabled (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{0,1}
资源使用	
说明	

2.11.20 LL_COMP_Lock

2.11.20.1 功能介绍

比较器锁定。

2.11.20.2 接口定义

函数接口	void LL_COMP_Lock (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	无
资源使用	
说明	

2.11.21 LL_COMP_IsLocked

2.11.21.1 功能介绍

获取比较器锁定状态。

2.11.21.2 接口定义

函数接口	uint32_t LL_COMP_IsLocked (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	{0,1}
资源使用	
说明	

2.11.22 LL_COMP_ReadOutputLevel

2.11.22.1 功能介绍

获取比较器输出电平。

2.11.22.2 接口定义

函数接口	uint32_t LL_COMP_ReadOutputLevel (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无

返回值	{ LL_COMP_OUTPUT_LEVEL_LOW, LL_COMP_OUTPUT_LEVEL_HIGH }
资源使用	
说明	

2.11.23 LL_COMP_DeInit

2.11.23.1 功能介绍

清除比较器初始化参数。

2.11.23.2 接口定义

函数接口	ErrorStatus LL_COMP_DeInit (COMP_TypeDef * COMPx)
输入	COMP_TypeDef * COMPx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.11.24 LL_COMP_Init

2.11.24.1 功能介绍

初始化比较器。

2.11.24.2 接口定义

函数接口	ErrorStatus LL_COMP_Init (COMP_TypeDef * COMPx, LL_COMP_InitTypeDef * COMP_InitStruct)
输入	COMP_TypeDef * COMPx LL_COMP_InitTypeDef * COMP_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.11.25 LL_COMP_StructInit

2.11.25.1 功能介绍

比较器结构体初始化。

2.11.25.2 接口定义

函数接口	void LL_COMP_StructInit (LL_COMP_InitTypeDef * COMP_InitStruct)
输入	LL_COMP_InitTypeDef * COMP_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.12 CRC 模块

属性	类型	字段名	含义
CRC_TypeDef			
读写	uint32_t	CR	控制状态寄存器
读写	uint32_t	RR	结果寄存器
读写	uint32_t	DR	数据寄存器
LL_CRC_InitTypeDef			
读写	uint32_t	CRCLength	CRC 长度

2.12.1 LL_CRC_SetPolynomialSize

2.12.1.1 功能介绍

配置多项式的大小。

2.12.1.2 接口定义

函数接口	void LL_CRC_SetPolynomialSize (CRC_TypeDef * CRCx, uint32_t PolySize)
输入	CRC_TypeDef * CRCx uint32_t PolySize: { LL_CRC_POLYLENGTH_32B, LL_CRC_POLYLENGTH_16B }
输出	无
返回值	无
资源使用	
说明	

2.12.2 LL_CRC_GetPolynomialSize

2.12.2.1 功能介绍

获取多项式的大小。

2.12.2.2 接口定义

函数接口	uint32_t LL_CRC_GetPolynomialSize (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	{ LL_CRC_POLYLENGTH_32B, LL_CRC_POLYLENGTH_16B }
资源使用	
说明	

2.12.3 LL_CRC_SetInitialData

2.12.3.1 功能介绍

初始化可编程 CRC 初始值。

2.12.3.2 接口定义

函数接口	void LL_CRC_SetInitialData (CRC_TypeDef * CRCx, uint32_t InitCrc)
------	---

输入	CRC_TypeDef * CRCx uint32_t InitCrc
输出	无
返回值	无
资源使用	
说明	

2.12.4 LL_CRC_GetInitialData

2.12.4.1 功能介绍

获取初始化可编程 CRC 初始值。

2.12.4.2 接口定义

函数接口	uint32_t LL_CRC_GetInitialData (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	初始值
资源使用	
说明	

2.12.5 LL_CRC_FeedData32

2.12.5.1 功能介绍

将给定的 32 位数据写入 CRC 计算器。

2.12.5.2 接口定义

函数接口	void LL_CRC_FeedData32 (CRC_TypeDef * CRCx, uint32_t InData)
输入	CRC_TypeDef * CRCx uint32_t InData: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.12.6 LL_CRC_FeedData16

2.12.6.1 功能介绍

将给定的 16 位数据写入 CRC 计算器。

2.12.6.2 接口定义

函数接口	void LL_CRC_FeedData16 (CRC_TypeDef * CRCx, uint16_t InData)
输入	CRC_TypeDef * CRCx uint16_t InData: [0x0000, 0xFFFF]
输出	无
返回值	无

资源使用	
说明	

2.12.7 LL_CRC_FeedData8

2.12.7.1 功能介绍

将给定的 8 位数据写入 CRC 计算器。

2.12.7.2 接口定义

函数接口	void LL_CRC_FeedData8(CRC_TypeDef * CRCx, uint8_t InData)
输入	CRC_TypeDef * CRCx uint8_t InData: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.12.8 LL_CRC_ReadData32

2.12.8.1 功能介绍

读取 CRC 32 位计算结果。

2.12.8.2 接口定义

函数接口	uint32_t LL_CRC_ReadData32 (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.12.9 LL_CRC_ReadData16

2.12.9.1 功能介绍

读取 CRC 16 位计算结果。

2.12.9.2 接口定义

函数接口	uint16_t LL_CRC_ReadData16 (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	[0x0000, 0xFFFF]
资源使用	
说明	

2.12.10 LL_CRC_ReadData8

2.12.10.1 功能介绍

读取 CRC 8 位计算结果。

2.12.10.2 接口定义

函数接口	uint8_t LL_CRC_ReadData8 (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.12.11LL_CRC_IsActiveFlag_ResultOK

2.12.11.1 功能介绍

获取当前 CRC 校验结果是否正确。

2.12.11.2 接口定义

函数接口	uint32_t LL_CRC_IsActiveFlag_ResultOK (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	{0,1}
资源使用	
说明	

2.12.12LL_CRC_DeInit

2.12.12.1 功能介绍

清除 CRC 初始化参数。

2.12.12.2 接口定义

函数接口	ErrorStatus LL_CRC_DeInit (CRC_TypeDef * CRCx)
输入	CRC_TypeDef * CRCx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.12.13LL_CRC_Init

2.12.13.1 功能介绍

CRC 初始化。

2.12.13.2 接口定义

函数接口	ErrorStatus LL_CRC_Init (CRC_TypeDef * CRCx, LL_CRC_InitTypeDef * CRC_InitStruct)
输入	CRC_TypeDef * CRCx LL_CRC_InitTypeDef * CRC_InitStruct
输出	无
返回值	ErrorStatus
资源使用	

说明	
----	--

2.12.14 LL_CRC_StructInit

2.12.14.1 功能介绍

CRC 结构体初始化。

2.12.14.2 接口定义

函数接口	void LL_CRC_StructInit (LL_CRC_InitTypeDef * CRC_InitStruct)
输入	LL_CRC_InitTypeDef * CRC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.12.15 LL_CRC_Accumulate

2.12.15.1 功能介绍

以先前计算的 CRC 作为初始值开始，计算 8 位数据缓冲区的 8、16 或 32 位 CRC 值。

2.12.15.2 接口定义

函数接口	uint32_t LL_CRC_Accumulate (CRC_TypeDef * CRCx, uint8_t pBuffer[], uint32_t BufferLength)
输入	CRC_TypeDef * CRCx uint8_t pBuffer[] uint32_t BufferLength
输出	无
返回值	CRC 计算值
资源使用	
说明	

2.12.16 LL_CRC_Calculate

2.12.16.1 功能介绍

从初始化值开始计算 8 位数据缓冲区的 8、16 或 32 位 CRC 值。

2.12.16.2 接口定义

函数接口	uint32_t LL_CRC_Calculate (CRC_TypeDef * CRCx, uint8_t pBuffer[], uint32_t BufferLength)
输入	CRC_TypeDef * CRCx uint8_t pBuffer[] uint32_t BufferLength
输出	无
返回值	CRC 计算值
资源使用	
说明	

2.13 DAC 模块

属性	类型	字段名	含义
DAC_TypeDef			
读写	uint32_t	CR	控制寄存器
只写	uint32_t	SWTRIGR	软件触发 寄存器
读写	uint32_t	DHR12R1	DAC1 通道 12 位 右对齐数据保持 寄存器
读写	uint32_t	DHR12L1	DAC1 通道 12 位 左对齐数据保持 寄存器
读写	uint32_t	DHR8R1	DAC1 通道 8 位右 对齐数据保持寄 存器
读写	uint32_t	DHR12R2	DAC2 通道 12 位 右对齐数据保持 寄存器
读写	uint32_t	DHR12L2	DAC2 通道 12 位 左对齐数据保持 寄存器
读写	uint32_t	DHR8R2	DAC2 通道 8 位右 对齐数据保持寄 存器
读写	uint32_t	DHR12RD	DAC 双通道 12 位 右对齐数据保持 寄存器
读写	uint32_t	DHR12LD	DAC 双通道 12 位 左对齐数据保持 寄存器
读写	uint32_t	DHR8RD	DAC 双通道 8 位 右对齐数据保持 寄存器
只读	uint32_t	DOR1	DAC1 通道数据输 出寄存器
只读	uint32_t	DOR2	DAC2 通道数据输

			出寄存器
读写	uint32_t	SR	状态寄存器
读写	uint32_t	MCR	模式控制寄存器
读写	uint32_t	CR2	控制寄存器 2
只写	uint32_t	SWTR2	软件触发寄存器 2
读写	uint32_t	DHR12R3	DAC3 通道 12 位 右对齐数据保持 寄存器
读写	uint32_t	DHR12R4	DAC4 通道 12 位 右对齐数据保持 寄存器
只读	uint32_t	DOR3	DAC3 通道数据输出 寄存器
只读	uint32_t	DOR4	DAC4 通道数据输出 寄存器
读写	uint32_t	SR2	状态寄存器 2
读写	uint32_t	MCR2	模式控制寄存器 2
LL_DAC_InitTypeDef			
读写	uint32_t	TriggerSource	触发源
读写	uint32_t	WaveAutoGeneration	波形自动生成模式
读写	uint32_t	WaveAutoGenerationConfig	波形自动生成配置
读写	uint32_t	OutputConnection	输出连接
读写	uint32_t	OutputBuffer	输出缓冲区
读写	uint32_t	OutputBufferMode	输出缓冲模式
读写	uint32_t	OutputBufferCalSrc	输出缓冲校准
读写	uint32_t	OutputBufferGain	输出缓冲增益

2.13.1 LL_DAC_SetSVREF

2.13.1.1 功能介绍

设置 DAC 通道参考电压。

2.13.1.2 接口定义

函数接口	void LL_DAC_SetSVREF (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Vref)
------	---

输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Vref: { LL_DAC_SVREF_AVDD, LL_DAC_SVREF_VREF }
输出	无
返回值	无
资源使用	
说明	

2.13.2 LL_DAC_GetSVREF

2.13.2.1 功能介绍

获取 DAC 通道参考电压。

2.13.2.2 接口定义

函数接口	uint32_t LL_DAC_GetSVREF (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_SVREF_AVDD, LL_DAC_SVREF_VREF }
资源使用	
说明	

2.13.3 LL_DAC_SetTriggerSource

2.13.3.1 功能介绍

设置 DAC 通道参考电压。

2.13.3.2 接口定义

函数接口	void LL_DAC_SetTriggerSource (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t TriggerSource)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t TriggerSource: { LL_DAC_TRIG_SOFTWARE, LL_DAC_TRIG_EXT_TIM1_TRGO, LL_DAC_TRIG_EXT_TIM2_TRGO, LL_DAC_TRIG_EXT_TIM3_TRGO, LL_DAC_TRIG_EXT_TIM4_TRGO, LL_DAC_TRIG_EXT_TIM5_TRGO, LL_DAC_TRIG_EXT_TIM6_TRGO, LL_DAC_TRIG_EXT_LPTIM1_OUT,

	LL_DAC_TRIG_EXT_EXTI_LINE9 }
输出	无
返回值	无
资源使用	
说明	

2.13.4 LL_DAC_GetTriggerSource

2.13.4.1 功能介绍

获取 DAC 通道参考电压。

2.13.4.2 接口定义

函数接口	uint32_t LL_DAC_GetTriggerSource (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_TRIG_SOFTWARE, LL_DAC_TRIG_EXT_TIM1_TRGO, LL_DAC_TRIG_EXT_TIM2_TRGO, LL_DAC_TRIG_EXT_TIM3_TRGO, LL_DAC_TRIG_EXT_TIM4_TRGO, LL_DAC_TRIG_EXT_TIM5_TRGO, LL_DAC_TRIG_EXT_TIM6_TRGO, LL_DAC_TRIG_EXT_LPTIM1_OUT, LL_DAC_TRIG_EXT_EXTI_LINE9 }
资源使用	
说明	

2.13.5 LL_DAC_SetWaveAutoGeneration

2.13.5.1 功能介绍

设置 DAC 通道的波形自动生成模式。

2.13.5.2 接口定义

函数接口	void LL_DAC_SetWaveAutoGeneration (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t WaveAutoGeneration)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t WaveAutoGeneration: { LL_DAC_WAVE_AUTO_GENERATION_NONE, LL_DAC_WAVE_AUTO_GENERATION_NOISE, LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE }
输出	无

返回值	无
资源使用	
说明	

2.13.6 LL_DAC_GetWaveAutoGeneration

2.13.6.1 功能介绍

获取 DAC 通道的波形自动生成模式。

2.13.6.2 接口定义

函数接口	uint32_t LL_DAC_GetWaveAutoGeneration (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_WAVE_AUTO_GENERATION_NONE, LL_DAC_WAVE_AUTO_GENERATION_NOISE, LL_DAC_WAVE_AUTO_GENERATION_TRIANGLE }
资源使用	
说明	

2.13.7 LL_DAC_SetWaveNoiseLFSR

2.13.7.1 功能介绍

设置 DAC 通道的噪声波形生成:噪声模式和参数 LFSR。

2.13.7.2 接口定义

函数接口	void LL_DAC_SetWaveNoiseLFSR (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t NoiseLFSRMask)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t NoiseLFSRMask: { LL_DAC_NOISE_LFSR_UNMASK_BIT0, LL_DAC_NOISE_LFSR_UNMASK_BITS1_0, LL_DAC_NOISE_LFSR_UNMASK_BITS2_0, LL_DAC_NOISE_LFSR_UNMASK_BITS3_0, LL_DAC_NOISE_LFSR_UNMASK_BITS4_0, LL_DAC_NOISE_LFSR_UNMASK_BITS5_0, LL_DAC_NOISE_LFSR_UNMASK_BITS6_0, LL_DAC_NOISE_LFSR_UNMASK_BITS7_0, LL_DAC_NOISE_LFSR_UNMASK_BITS8_0, LL_DAC_NOISE_LFSR_UNMASK_BITS9_0, LL_DAC_NOISE_LFSR_UNMASK_BITS10_0, LL_DAC_NOISE_LFSR_UNMASK_BITS11_0 }

输出	无
返回值	无
资源使用	
说明	

2.13.8 LL_DAC_GetWaveNoiseLFSR

2.13.8.1 功能介绍

获取 DAC 通道的噪声波形生成:噪声模式和参数 LFSR。

2.13.8.2 接口定义

函数接口	uint32_t LL_DAC_GetWaveNoiseLFSR (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_NOISE_LFSR_UNMASK_BIT0, LL_DAC_NOISE_LFSR_UNMASK_BITS1_0, LL_DAC_NOISE_LFSR_UNMASK_BITS2_0, LL_DAC_NOISE_LFSR_UNMASK_BITS3_0, LL_DAC_NOISE_LFSR_UNMASK_BITS4_0, LL_DAC_NOISE_LFSR_UNMASK_BITS5_0, LL_DAC_NOISE_LFSR_UNMASK_BITS6_0, LL_DAC_NOISE_LFSR_UNMASK_BITS7_0, LL_DAC_NOISE_LFSR_UNMASK_BITS8_0, LL_DAC_NOISE_LFSR_UNMASK_BITS9_0, LL_DAC_NOISE_LFSR_UNMASK_BITS10_0, LL_DAC_NOISE_LFSR_UNMASK_BITS11_0 }
资源使用	
说明	

2.13.9 LL_DAC_SetWaveTriangleAmplitude

2.13.9.1 功能介绍

设置 DAC 通道的三角波形生成:三角模式和振幅。

2.13.9.2 接口定义

函数接口	void LL_DAC_SetWaveTriangleAmplitude (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t TriangleAmplitude)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t TriangleAmplitude: { LL_DAC_TRIANGLE_AMPLITUDE_1, LL_DAC_TRIANGLE_AMPLITUDE_3,

	LL_DAC_TRIANGLE_AMPLITUDE_7, LL_DAC_TRIANGLE_AMPLITUDE_15, LL_DAC_TRIANGLE_AMPLITUDE_31, LL_DAC_TRIANGLE_AMPLITUDE_63, LL_DAC_TRIANGLE_AMPLITUDE_127, LL_DAC_TRIANGLE_AMPLITUDE_255, LL_DAC_TRIANGLE_AMPLITUDE_511, LL_DAC_TRIANGLE_AMPLITUDE_1023, LL_DAC_TRIANGLE_AMPLITUDE_2047, LL_DAC_TRIANGLE_AMPLITUDE_4095 }
输出	无
返回值	无
资源使用	
说明	

2.13.10 LL_DAC_GetWaveTriangleAmplitude

2.13.10.1 功能介绍

获取 DAC 通道的三角波形生成:三角模式和振幅。

2.13.10.2 接口定义

函数接口	uint32_t LL_DAC_GetWaveTriangleAmplitude (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_TRIANGLE_AMPLITUDE_1, LL_DAC_TRIANGLE_AMPLITUDE_3, LL_DAC_TRIANGLE_AMPLITUDE_7, LL_DAC_TRIANGLE_AMPLITUDE_15, LL_DAC_TRIANGLE_AMPLITUDE_31, LL_DAC_TRIANGLE_AMPLITUDE_63, LL_DAC_TRIANGLE_AMPLITUDE_127, LL_DAC_TRIANGLE_AMPLITUDE_255, LL_DAC_TRIANGLE_AMPLITUDE_511, LL_DAC_TRIANGLE_AMPLITUDE_1023, LL_DAC_TRIANGLE_AMPLITUDE_2047, LL_DAC_TRIANGLE_AMPLITUDE_4095 }
资源使用	
说明	

2.13.11 LL_DAC_SetOutputBuffer

2.13.11.1 功能介绍

设置 DAC 通道的输出缓冲区。

2.13.11.2 接口定义

函数接口	void LL_DAC_SetOutputBuffer (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t OutputBuffer)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t OutputBuffer: { LL_DAC_OUTPUT_BUFFER_ENABLE, LL_DAC_OUTPUT_BUFFER_DISABLE }
输出	无
返回值	无
资源使用	
说明	

2.13.12 LL_DAC_GetOutputBuffer

2.13.12.1 功能介绍

获取 DAC 通道的输出缓冲区。

2.13.12.2 接口定义

函数接口	uint32_t LL_DAC_GetOutputBuffer (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_OUTPUT_BUFFER_ENABLE, LL_DAC_OUTPUT_BUFFER_DISABLE }
资源使用	
说明	

2.13.13 LL_DAC_SetOutputConnection

2.13.13.1 功能介绍

设置 DAC 通道的输出连接。

2.13.13.2 接口定义

函数接口	void LL_DAC_SetOutputConnection (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t OutputConnection)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t OutputConnection: { LL_DAC_OUTPUT_CONNECT_INTERNAL,

	LL_DAC_OUTPUT_CONNECT_IO1, LL_DAC_OUTPUT_CONNECT_IO2, LL_DAC_OUTPUT_CONNECT_IO3 }
输出	无
返回值	无
资源使用	
说明	

2.13.14 LL_DAC_GetOutputConnection

2.13.14.1 功能介绍

获取 DAC 通道的输出连接。

2.13.14.2 接口定义

函数接口	uint32_t LL_DAC_GetOutputConnection (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_OUTPUT_CONNECT_INTERNAL, LL_DAC_OUTPUT_CONNECT_IO1, LL_DAC_OUTPUT_CONNECT_IO2, LL_DAC_OUTPUT_CONNECT_IO3 }
资源使用	
说明	

2.13.15 LL_DAC_SetOutputBufferMode

2.13.15.1 功能介绍

设置 DAC 通道的输出缓冲模式。

2.13.15.2 接口定义

函数接口	void LL_DAC_SetOutputBufferMode (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t BufferMode)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t BufferMode: { LL_DAC_OUTPUT_BUFFERMODE_NORMAL, LL_DAC_OUTPUT_BUFFERMODE_CAL }
输出	无
返回值	无
资源使用	
说明	

2.13.16 LL_DAC_GetOutputBufferMode

2.13.16.1 功能介绍

获取 DAC 通道的输出缓冲模式。

2.13.16.2 接口定义

函数接口	uint32_t LL_DAC_GetOutputBufferMode (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_OUTPUT_BUFFERMODE_NORMAL, LL_DAC_OUTPUT_BUFFERMODE_CAL }
资源使用	
说明	

2.13.17 LL_DAC_SetOutputBufferCalibrationLSI

2.13.17.1 功能介绍

设置 DAC 通道的输出缓冲校准源。

2.13.17.2 接口定义

函数接口	void LL_DAC_SetOutputBufferCalibrationLSI (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t BufferCalLSI)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t BufferCalLSI: { LL_DAC_OUTPUT_BUFFER_CALLSI_VSS, LL_DAC_OUTPUT_BUFFER_CALLSI_VBG }
输出	无
返回值	无
资源使用	
说明	

2.13.18 LL_DAC_GetOutputBufferCalibrationLSI

2.13.18.1 功能介绍

获取 DAC 通道的输出缓冲校准源。

2.13.18.2 接口定义

函数接口	uint32_t LL_DAC_GetOutputBufferCalibrationLSI (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2,

	LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_OUTPUT_BUFFER_CALLSI_VSS, LL_DAC_OUTPUT_BUFFER_CALLSI_VBG }
资源使用	
说明	

2.13.19 LL_DAC_SetOutputBufferGain

2.13.19.1 功能介绍

设置 DAC 通道的输出缓冲增益。

2.13.19.2 接口定义

函数接口	void LL_DAC_SetOutputBufferGain (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t BufferGain)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t BufferGain: { LL_DAC_OUTPUT_BUFFER_GAIN_1, LL_DAC_OUTPUT_BUFFER_GAIN_100 }
输出	无
返回值	无
资源使用	
说明	

2.13.20 LL_DAC_GetOutputBufferGain

2.13.20.1 功能介绍

获取 DAC 通道的输出缓冲增益。

2.13.20.2 接口定义

函数接口	uint32_t LL_DAC_GetOutputBufferGain (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{ LL_DAC_OUTPUT_BUFFER_GAIN_1, LL_DAC_OUTPUT_BUFFER_GAIN_100 }
资源使用	
说明	

2.13.21 LL_DAC_EnableDMAReq

2.13.21.1 功能介绍

使能 DAC 通道 DMA 传输请求。

2.13.21.2 接口定义

函数接口	void LL_DAC_EnableDMAReq (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.22 LL_DAC_DisableDMAReq

2.13.22.1 功能介绍

禁止使能 DAC 通道 DMA 传输请求。

2.13.22.2 接口定义

函数接口	void LL_DAC_DisableDMAReq (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.23 LL_DAC_IsDMAReqEnabled

2.13.23.1 功能介绍

检查是否使能 DAC 通道 DMA 传输请求。

2.13.23.2 接口定义

函数接口	uint32_t LL_DAC_IsDMAReqEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.13.24 LL_DAC_DMA_GetRegAddr

2.13.24.1 功能介绍

用于 DMA 传输的 DAC 寄存器地址。

2.13.24.2 接口定义

函数接口	uint32_t LL_DAC_DMA_GetRegAddr (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Register)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Register: { LL_DAC_DMA_REG_DATA_12BITS_RIGHT_ALIGNED, LL_DAC_DMA_REG_DATA_12BITS_LEFT_ALIGNED, LL_DAC_DMA_REG_DATA_8BITS_RIGHT_ALIGNED }
输出	无
返回值	寄存器地址
资源使用	
说明	

2.13.25 LL_DAC_Enable

2.13.25.1 功能介绍

使能 DAC 通道。

2.13.25.2 接口定义

函数接口	void LL_DAC_Enable (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.26 LL_DAC_Disable

2.13.26.1 功能介绍

禁止使能 DAC 通道。

2.13.26.2 接口定义

函数接口	void LL_DAC_Disable(DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }

输出	无
返回值	无
资源使用	
说明	

2.13.27 LL_DAC_IsEnabled

2.13.27.1 功能介绍

检查是否使能 DAC 通道。

2.13.27.2 接口定义

函数接口	uint32_t LL_DAC_IsEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.13.28 LL_DAC_EnableTrigger

2.13.28.1 功能介绍

使能 DAC 通道触发。

2.13.28.2 接口定义

函数接口	void LL_DAC_EnableTrigger (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.29 LL_DAC_DisableTrigger

2.13.29.1 功能介绍

禁止使能 DAC 通道触发。

2.13.29.2 接口定义

函数接口	void LL_DAC_DisableTrigger (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel:

	{ LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.30 LL_DAC_IsEnabledTrigger

2.13.30.1 功能介绍

检查是否使能 DAC 通道触发。

2.13.30.2 接口定义

函数接口	uint32_t LL_DAC_IsTriggerEnabled (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.13.31 LL_DAC_TrigSWConversion

2.13.31.1 功能介绍

设置 DAC 通道软件触发转换。

2.13.31.2 接口定义

函数接口	void LL_DAC_TrigSWConversion (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.32 LL_DAC_ConvertData12RightAligned

2.13.32.1 功能介绍

设置数据保存寄存器中要加载的数据，格式为所选 DAC 通道右对齐 12 位。

2.13.32.2 接口定义

函数接口	void LL_DAC_ConvertData12RightAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)
------	---

输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Data: [0x000, 0xFFFF]
输出	无
返回值	无
资源使用	
说明	

2.13.33 LL_DAC_ConvertData12LeftAligned

2.13.33.1 功能介绍

设置数据保存寄存器中要加载的数据，格式为所选 DAC 通道左对齐 12 位。

2.13.33.2 接口定义

函数接口	void LL_DAC_ConvertData12LeftAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2 } uint32_t Data: [0x0000, 0xFFFF0]
输出	无
返回值	无
资源使用	
说明	

2.13.34 LL_DAC_ConvertData8RightAligned

2.13.34.1 功能介绍

设置数据保存寄存器中要加载的数据，格式为所选 DAC 通道右对齐 8 位。

2.13.34.2 接口定义

函数接口	void LL_DAC_ConvertData8RightAligned (DAC_TypeDef * DACx, uint32_t DAC_Channel, uint32_t Data)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2 } uint32_t Data: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.13.35 LL_DAC_ConvertDualData12RightAligned

2.13.35.1 功能介绍

设置数据加载在数据保存寄存器格式为 12 位右对齐的两个 DAC 通道。

2.13.35.2 接口定义

函数接口	void LL_DAC_ConvertDualData12RightAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)
输入	DAC_TypeDef * DACx uint32_t DataChannel1: [0x000, 0xFFF] uint32_t DataChannel2: [0x000, 0xFFF]
输出	无
返回值	无
资源使用	
说明	

2.13.36 LL_DAC_ConvertDualData12LeftAligned

2.13.36.1 功能介绍

设置数据加载在数据保存寄存器格式为 12 位左对齐的两个 DAC 通道。

2.13.36.2 接口定义

函数接口	void LL_DAC_ConvertDualData12LeftAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)
输入	DAC_TypeDef * DACx uint32_t DataChannel1: [0x000, 0xFFF] uint32_t DataChannel2: [0x0000, 0xFFF0]
输出	无
返回值	无
资源使用	
说明	

2.13.37 LL_DAC_ConvertDualData8RightAligned

2.13.37.1 功能介绍

设置数据加载在数据保存寄存器格式为 8 位右对齐的两个 DAC 通道。

2.13.37.2 接口定义

函数接口	void LL_DAC_ConvertDualData8RightAligned (DAC_TypeDef * DACx, uint32_t DataChannel1, uint32_t DataChannel2)
输入	DAC_TypeDef * DACx uint32_t DataChannel1: [0x00, 0xFF] uint32_t DataChannel2:

	[0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.13.38 LL_DAC_RetrieveOutputData

2.13.38.1 功能介绍

获取当前为 DAC 通道生成的输出数据。

2.13.38.2 接口定义

函数接口	uint32_t LL_DAC_RetrieveOutputData (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	[0x000, 0xFFFF]
资源使用	
说明	

2.13.39 LL_DAC_IsActiveFlag_DMAUDR

2.13.39.1 功能介绍

获取 DAC 通道溢出标志。

2.13.39.2 接口定义

函数接口	uint32_t LL_DAC_IsActiveFlag_DMAUDR (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.13.40 LL_DAC_ClearFlag_DMAUDR

2.13.40.1 功能介绍

清除 DAC 通道溢出标志。

2.13.40.2 接口定义

函数接口	void LL_DAC_ClearFlag_DMAUDR (DAC_TypeDef * DACx, uint32_t DAC_Channel)
------	---

输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.41 LL_DAC_EnableIT_DMAUDR

2.13.41.1 功能介绍

使能 DAC 通道 DMA 溢出中断。

2.13.41.2 接口定义

函数接口	void LL_DAC_EnableIT_DMAUDR (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.42 LL_DAC_DisableIT_DMAUDR

2.13.42.1 功能介绍

禁止使能 DAC 通道 DMA 溢出中断。

2.13.42.2 接口定义

函数接口	void LL_DAC_DisableIT_DMAUDR (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

2.13.43 LL_DAC_IsEnabledIT_DMAUDR

2.13.43.1 功能介绍

检查是否使能 DAC 通道 DMA 溢出中断。

2.13.43.2 接口定义

函数接口	uint32_t LL_DAC_IsEnabledIT_DMAUDR (DAC_TypeDef * DACx, uint32_t DAC_Channel)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.13.44 LL_DAC_DeInit

2.13.44.1 功能介绍

清除 DAC 初始化参数。

2.13.44.2 接口定义

函数接口	ErrorStatus LL_DAC_DeInit (DAC_TypeDef * DACx)
输入	DAC_TypeDef * DACx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.13.45 LL_DAC_Init

2.13.45.1 功能介绍

DAC 初始化。

2.13.45.2 接口定义

函数接口	ErrorStatus LL_DAC_Init (DAC_TypeDef * DACx, uint32_t DAC_Channel, LL_DAC_InitTypeDef * DAC_InitStruct)
输入	DAC_TypeDef * DACx uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } LL_DAC_InitTypeDef * DAC_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.13.46 LL_DAC_StructInit

2.13.46.1 功能介绍

DAC 结构体初始化。

2.13.46.2 接口定义

函数接口	void LL_DAC_StructInit (LL_DAC_InitTypeDef * DAC_InitStruct)
输入	LL_DAC_InitTypeDef * DAC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.14 DMA 模块

属性	类型	字段名	含义
DMA_TypeDef			
只读	uint32_t	ISR	中断状态寄存器
只写	uint32_t	ICR	中断标志清除寄存器
读写	uint32_t	CCR0	DMA 通道 0 控制寄存器
读写	uint32_t	CNDTR0	DMA 通道 0 待传输次数寄存器
读写	uint32_t	CSAR0	DMA 通道 0 源地址寄存器
读写	uint32_t	CDAR0	DMA 通道 0 目的地址寄存器
读写	uint32_t	CCR1	DMA 通道 1 控制寄存器
读写	uint32_t	CNDTR1	DMA 通道 1 待传输次数寄存器
读写	uint32_t	CSAR1	DMA 通道 1 源地址寄存器
读写	uint32_t	CDAR1	DMA 通道 1 目的地址寄存器
读写	uint32_t	CCR2	DMA 通道 2 控制寄存器
只读	uint32_t	CNDTR2	DMA 通道 2 待传输次数寄存器
只读	uint32_t	CSAR2	DMA 通道 2 源地址寄存器

读写	uint32_t	CDAR2	DMA 通道 2 目的地址寄存器
读写	uint32_t	CCR3	DMA 通道 3 控制寄存器
读写	uint32_t	CNDTR3	DMA 通道 3 待传输次数寄存器
只写	uint32_t	CSAR3	DMA 通道 3 源地址寄存器
读写	uint32_t	CDAR3	DMA 通道 3 目的地址寄存器
LL_DMA_InitTypeDef			
读写	uint32_t	SrcAddress	源地址
读写	uint32_t	DstAddress	目的地址
读写	uint32_t	SoftwareRequire	软件触发请求
读写	uint32_t	Mode	操作模式
读写	uint32_t	SrcIncMode	源地址递增模式
读写	uint32_t	DstIncMode	目的地址递增模式
读写	uint32_t	DataSize	传输数据大小
读写	uint32_t	NbData	待传输次数
读写	uint32_t	PeriphRequest	外设请求 ID
读写	uint32_t	Priority	优先级
DMAMUX_TypeDef			
读写	uint32_t	C0CR	请求复用器通道 0 控制寄存器
读写	uint32_t	C1CR	请求复用器通道 1 控制寄存器
读写	uint32_t	C2CR	请求复用器通道 2 控制寄存器
读写	uint32_t	C3CR	请求复用器通道 3 控制寄存器
读写	uint32_t	RG0CR	请求生成器 0 控制寄存器
读写	uint32_t	RG1CR	请求生成器 1 控制寄存器

只读	uint32_t	RGSR	请求生成器中断状态寄存器
只写	uint32_t	RGCFR	请求生成器中断标志清除寄存器

2.14.1 LL_DMA_EnableChannel

2.14.1.1 功能介绍

使能 DMA 通道。

2.14.1.2 接口定义

函数接口	void LL_DMA_EnableChannel (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.2 LL_DMA_DisableChannel

2.14.2.1 功能介绍

禁止使能 DMA 通道。

2.14.2.2 接口定义

函数接口	void LL_DMA_DisableChannel (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.3 LL_DMA_IsEnabledChannel

2.14.3.1 功能介绍

检查是否使能 DMA 通道。

2.14.3.2 接口定义

函数接口	uint32_t LL_DMA_IsEnabledChannel (DMA_TypeDef * DMAx, uint32_t Channel)
------	---

输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.4 LL_DMA_SetSoftwareRequire

2.14.4.1 功能介绍

设置 DMA 通道软件请求触发是否使能。

2.14.4.2 接口定义

函数接口	void LL_DMA_SetSoftwareRequire (DMA_TypeDef * DMAx,uint32_t Channel, uint32_t SoftwareRequire)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t SoftwareRequire: { LL_DMA_SOFTWARE_REQUIRE_DISABLE, LL_DMA_SOFTWARE_REQUIRE_ENABLE }
输出	无
返回值	无
资源使用	
说明	

2.14.5 LL_DMA_GetSoftwareRequire

2.14.5.1 功能介绍

获取 DMA 通道软件请求触发是否使能。

2.14.5.2 接口定义

函数接口	uint32_t LL_DMA_GetSoftwareRequire (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{ LL_DMA_SOFTWARE_REQUIRE_DISABLE, LL_DMA_SOFTWARE_REQUIRE_ENABLE }
资源使用	
说明	

2.14.6 LL_DMA_ConfigTransfer

2.14.6.1 功能介绍

配置所有参数链接到 DMA 传输。

2.14.6.2 接口定义

函数接口	void LL_DMA_ConfigTransfer (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t Configuration)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t Configuration: { LL_DMA_MODE_NORMAL, LL_DMA_MODE_CIRCULAR, LL_DMA_SOFTWARE_REQUIRE_DISABLE, LL_DMA_SOFTWARE_REQUIRE_ENABLE, LL_DMA_SOURCE_INCREMENT, LL_DMA_SOURCE_NOINCREMENT, LL_DMA_DESTINATION_INCREMENT, LL_DMA_DESTINATION_NOINCREMENT, LL_DMA_DATAALIGN_BYTE, LL_DMA_DATAALIGN_HALFWORD, LL_DMA_DATAALIGN_WORD, LL_DMA_PRIORITY_LOW, LL_DMA_PRIORITY_MEDIUM, LL_DMA_PRIORITY_HIGH, LL_DMA_PRIORITY_VERYHIGH }
输出	无
返回值	无
资源使用	
说明	

2.14.7 LL_DMA_SetMode

2.14.7.1 功能介绍

设置 DMA 模式为循环模式或正常模式。

2.14.7.2 接口定义

函数接口	void LL_DMA_SetMode (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t Mode)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t Mode: { LL_DMA_MODE_NORMAL, LL_DMA_MODE_CIRCULAR}
输出	无
返回值	无
资源使用	

说明	
----	--

2.14.8 LL_DMA_GetMode

2.14.8.1 功能介绍

获取 DMA 模式为循环模式或正常模式。

2.14.8.2 接口定义

函数接口	uint32_t LL_DMA_GetMode (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{ LL_DMA_MODE_NORMAL, LL_DMA_MODE_CIRCULAR }
资源使用	
说明	

2.14.9 LL_DMA_SetSourceIncMode

2.14.9.1 功能介绍

设置源地址递增模式。

2.14.9.2 接口定义

函数接口	void LL_DMA_SetSourceIncMode (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t LSIIncMode)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t LSIIncMode: { LL_DMA_SOURCE_INCREMENT, LL_DMA_SOURCE_NOINCREMENT }
输出	无
返回值	无
资源使用	
说明	

2.14.10 LL_DMA_GetSourceIncMode

2.14.10.1 功能介绍

获取源地址递增模式。

2.14.10.2 接口定义

函数接口	uint32_t LL_DMA_GetSourceIncMode (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel:

	{ LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{ LL_DMA_SOURCE_INCREMENT, LL_DMA_SOURCE_NOINCREMENT }
资源使用	
说明	

2.14.11 LL_DMA_SetDestinationIncMode

2.14.11.1 功能介绍

设置目的地址递增模式。

2.14.11.2 接口定义

函数接口	void LL_DMA_SetDestinationIncMode (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t DstIncMode)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t DstIncMode: { LL_DMA_DESTINATION_INCREMENT, LL_DMA_DESTINATION_NOINCREMENT }
输出	无
返回值	无
资源使用	
说明	

2.14.12 LL_DMA_GetDestinationIncMode

2.14.12.1 功能介绍

获取目的地址递增模式。

2.14.12.2 接口定义

函数接口	uint32_t LL_DMA_GetDestinationIncMode (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{ LL_DMA_DESTINATION_INCREMENT, LL_DMA_DESTINATION_NOINCREMENT }
资源使用	
说明	

2.14.13 LL_DMA_SetDataSize

2.14.13.1 功能介绍

设置传输数据位数。

2.14.13.2 接口定义

函数接口	void LL_DMA_SetDataSize (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t PeriphOrM2MLSIDataSize)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t PeriphOrM2MLSIDataSize: { LL_DMA_DATAALIGN_BYTE, LL_DMA_DATAALIGN_HALFWORD, LL_DMA_DATAALIGN_WORD }
输出	无
返回值	无
资源使用	
说明	

2.14.14 LL_DMA_GetDataSize

2.14.14.1 功能介绍

获取传输数据位数。

2.14.14.2 接口定义

函数接口	uint32_t LL_DMA_GetDataSize (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{ LL_DMA_DATAALIGN_BYTE, LL_DMA_DATAALIGN_HALFWORD, LL_DMA_DATAALIGN_WORD }
资源使用	
说明	

2.14.15 LL_DMA_SetChannelPriorityLevel

2.14.15.1 功能介绍

设置通道优先级。

2.14.15.2 接口定义

函数接口	void LL_DMA_SetChannelPriorityLevel (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t Priority)
------	---

输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t Priority: { LL_DMA_PRIORITY_LOW, LL_DMA_PRIORITY_MEDIUM, LL_DMA_PRIORITY_HIGH, LL_DMA_PRIORITY_VERYHIGH }
输出	无
返回值	无
资源使用	
说明	

2.14.16 LL_DMA_GetChannelPriorityLevel

2.14.16.1 功能介绍

获取通道优先级。

2.14.16.2 接口定义

函数接口	uint32_t LL_DMA_GetChannelPriorityLevel (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{ LL_DMA_PRIORITY_LOW, LL_DMA_PRIORITY_MEDIUM, LL_DMA_PRIORITY_HIGH, LL_DMA_PRIORITY_VERYHIGH }
资源使用	
说明	

2.14.17 LL_DMA_SetDataLength

2.14.17.1 功能介绍

设置数据长度。

2.14.17.2 接口定义

函数接口	void LL_DMA_SetDataLength (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t NbData)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t NbData: [0x00000000, 0x0000FFFF]
输出	无
返回值	无
资源使用	

说明	
----	--

2.14.18 LL_DMA_GetDataLength

2.14.18.1 功能介绍

获取数据长度。

2.14.18.2 接口定义

函数接口	uint32_t LL_DMA_GetDataLength (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	[0x00000000, 0x0000FFFF]
资源使用	
说明	

2.14.19 LL_DMA_ConfigAddresses

2.14.19.1 功能介绍

配置源地址和目的地址。

2.14.19.2 接口定义

函数接口	void LL_DMA_ConfigAddresses (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t LSIAAddress, uint32_t DstAddress)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t LSIAAddress: [0x00000000, 0xFFFFFFFF] uint32_t DstAddress: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.14.20 LL_DMA_SetSourceAddress

2.14.20.1 功能介绍

配置源地址。

2.14.20.2 接口定义

函数接口	void LL_DMA_SetSourceAddress (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t SourceAddress)
输入	DMA_TypeDef * DMAx

	uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t SourceAddress: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.14.21 LL_DMA_SetDestinationAddress

2.14.21.1 功能介绍

配置目的地址。

2.14.21.2 接口定义

函数接口	void LL_DMA_SetDestinationAddress (DMA_TypeDef * DMAx, uint32_t Channel, uint32_t DstAddress)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } uint32_t DstAddress: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.14.22 LL_DMA_GetSourceAddress

2.14.22.1 功能介绍

获取源地址。

2.14.22.2 接口定义

函数接口	uint32_t LL_DMA_GetSourceAddress (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.14.23 LL_DMA_GetDestinationAddress

2.14.23.1 功能介绍

获取目的地址。

2.14.23.2 接口定义

函数接口	uint32_t LL_DMA_GetDestinationAddress (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.14.24 LL_DMA_IsActiveFlag_GI

2.14.24.1 功能介绍

检查 DMA 通道的全局中断标志。

2.14.24.2 接口定义

函数接口	uint32_t LL_DMA_IsActiveFlag_GI (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.25 LL_DMA_IsActiveFlag_TC

2.14.25.1 功能介绍

检查 DMA 通道的传输完成标志。

2.14.25.2 接口定义

函数接口	uint32_t LL_DMA_IsActiveFlag_TC (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}

资源使用	
说明	

2.14.26 LL_DMA_IsActiveFlag_HT

2.14.26.1 功能介绍

检查 DMA 通道的传输完成一半标志。

2.14.26.2 接口定义

函数接口	uint32_t LL_DMA_IsActiveFlag_HT (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.27 LL_DMA_IsActiveFlag_TE

2.14.27.1 功能介绍

检查 DMA 通道的传输错误标志。

2.14.27.2 接口定义

函数接口	uint32_t LL_DMA_IsActiveFlag_TE (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.28 LL_DMA_ClearFlag_GI

2.14.28.1 功能介绍

清除 DMA 通道的全局中断标志。

2.14.28.2 接口定义

函数接口	void LL_DMA_ClearFlag_GI (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }

输出	无
返回值	无
资源使用	
说明	

2.14.29 LL_DMA_ClearFlag_TC

2.14.29.1 功能介绍

清除 DMA 通道的传输完成标志。

2.14.29.2 接口定义

函数接口	void LL_DMA_ClearFlag_TC (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.30 LL_DMA_ClearFlag_HT

2.14.30.1 功能介绍

清除 DMA 通道的传输完成一半标志。

2.14.30.2 接口定义

函数接口	void LL_DMA_ClearFlag_HT (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.31 LL_DMA_ClearFlag_TE

2.14.31.1 功能介绍

清除 DMA 通道的传输错误标志。

2.14.31.2 接口定义

函数接口	void LL_DMA_ClearFlag_TE (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel:

	{ LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.32 LL_DMA_EnableIT_TC

2.14.32.1 功能介绍

使能 DMA 通道的传输完成中断。

2.14.32.2 接口定义

函数接口	void LL_DMA_EnableIT_TC (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.33 LL_DMA_EnableIT_HT

2.14.33.1 功能介绍

使能 DMA 通道的传输完成一半中断。

2.14.33.2 接口定义

函数接口	void LL_DMA_EnableIT_HT (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.34 LL_DMA_EnableIT_TE

2.14.34.1 功能介绍

使能 DMA 通道的传输错误中断。

2.14.34.2 接口定义

函数接口	void LL_DMA_EnableIT_TE (DMA_TypeDef * DMAx, uint32_t Channel)
------	--

输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.35 LL_DMA_DisableIT_TC

2.14.35.1 功能介绍

禁止使能 DMA 通道的传输完成中断。

2.14.35.2 接口定义

函数接口	void LL_DMA_DisableIT_TC (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.36 LL_DMA_DisableIT_HT

2.14.36.1 功能介绍

禁止使能 DMA 通道的传输完成一半中断。

2.14.36.2 接口定义

函数接口	void LL_DMA_DisableIT_HT (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.37 LL_DMA_DisableIT_TE

2.14.37.1 功能介绍

禁止使能 DMA 通道的传输错误中断。

2.14.37.2 接口定义

函数接口	void LL_DMA_DisableIT_TE (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.38 LL_DMA_IsEnabledIT_TC

2.14.38.1 功能介绍

检查是否使能 DMA 通道的传输完成中断。

2.14.38.2 接口定义

函数接口	uint32_t LL_DMA_IsEnabledIT_TC (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.39 LL_DMA_IsEnabledIT_HT

2.14.39.1 功能介绍

检查是否使能 DMA 通道的传输完成一半中断。

2.14.39.2 接口定义

函数接口	uint32_t LL_DMA_IsEnabledIT_HT (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.40 LL_DMA_IsEnabledIT_TE

2.14.40.1 功能介绍

检查是否使能 DMA 通道的传输错误中断。

2.14.40.2 接口定义

函数接口	uint32_t LL_DMA_IsEnabledIT_TE (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.41 LL_DMA_Init

2.14.41.1 功能介绍

初始化 DMA。

2.14.41.2 接口定义

函数接口	ErrorStatus LL_DMA_Init (DMA_TypeDef * DMAx, uint32_t Channel, LL_DMA_InitTypeDef * DMA_InitStruct)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 } LL_DMA_InitTypeDef * DMA_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.14.42 LL_DMA_DeInit

2.14.42.1 功能介绍

清除 DMA 初始化参数。

2.14.42.2 接口定义

函数接口	ErrorStatus LL_DMA_DeInit (DMA_TypeDef * DMAx, uint32_t Channel)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无

返回值	ErrorStatus
资源使用	
说明	

2.14.43 LL_DMA_StructInit

2.14.43.1 功能介绍

初始化 DMA 结构体。

2.14.43.2 接口定义

函数接口	void LL_DMA_StructInit (LL_DMA_InitTypeDef * DMA_InitStruct)
输入	DMA_TypeDef * DMAx uint32_t Channel: { LL_DMA_CHANNEL_0, LL_DMA_CHANNEL_1, LL_DMA_CHANNEL_2, LL_DMA_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.44 LL_DMAMUX_SetRequestID

2.14.44.1 功能介绍

设置 DMAMUX 请求信号源标识。

2.14.44.2 接口定义

函数接口	void LL_DMAMUX_SetRequestID (DMAMUX_TypeDef * DMAMUXx, uint32_t Channel, uint32_t Request)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1, LL_DMAMUX_CHANNEL_2, LL_DMAMUX_CHANNEL_3 } uint32_t Request: { LL_DMAMUX_REQ_MEM2MEM, LL_DMAMUX_REQ_GENERATOR0, LL_DMAMUX_REQ_GENERATOR1, LL_DMAMUX_REQ_ADC1, In Channel 0 or 2: LL_DMAMUX_REQ0_2_I2C1_RX, LL_DMAMUX_REQ0_2_I2C2_RX, LL_DMAMUX_REQ0_2_LPUART1_RX, LL_DMAMUX_REQ0_2_SPI1_RX, LL_DMAMUX_REQ0_2_SPI2_RX, LL_DMAMUX_REQ0_2_TIM1_CH1, LL_DMAMUX_REQ0_2_TIM1_CH3, LL_DMAMUX_REQ0_2_TIM1_TRIG_COM, LL_DMAMUX_REQ0_2_TIM2_CH1, LL_DMAMUX_REQ0_2_TIM2_CH3, LL_DMAMUX_REQ0_2_TIM2_TRIG, LL_DMAMUX_REQ0_2_TIM3_CH1, LL_DMAMUX_REQ0_2_TIM3_CH3,

	LL_DMAMUX_REQ0_2_TIM3_TRIG, LL_DMAMUX_REQ0_2_TIM4_CH1, LL_DMAMUX_REQ0_2_TIM4_CH3, LL_DMAMUX_REQ0_2_TIM4_TRIG, LL_DMAMUX_REQ0_2_TIM5_UP, LL_DMAMUX_REQ0_2_TIM7_UP, LL_DMAMUX_REQ0_2_USART1_RX, LL_DMAMUX_REQ0_2_USART2_RX, LL_DMAMUX_REQ0_2_USART3_RX, LL_DMAMUX_REQ0_2_DAC_CH1, LL_DMAMUX_REQ0_2_DAC_CH3, LL_DMAMUX_REQ0_2_ATK_A, In Channel 1 or 3 : LL_DMAMUX_REQ1_3_I2C1_TX, LL_DMAMUX_REQ1_3_I2C2_TX, LL_DMAMUX_REQ1_3_LPUART1_TX, LL_DMAMUX_REQ1_3_SPI1_TX, LL_DMAMUX_REQ1_3_SPI2_TX, LL_DMAMUX_REQ1_3_TIM1_CH2, LL_DMAMUX_REQ1_3_TIM1_CH4, LL_DMAMUX_REQ1_3_TIM1_UP, LL_DMAMUX_REQ1_3_TIM2_CH2, LL_DMAMUX_REQ1_3_TIM2_CH4, LL_DMAMUX_REQ1_3_TIM2_UP, LL_DMAMUX_REQ1_3_TIM3_CH2, LL_DMAMUX_REQ1_3_TIM3_CH4, LL_DMAMUX_REQ1_3_TIM3_UP, LL_DMAMUX_REQ1_3_TIM4_CH2, LL_DMAMUX_REQ1_3_TIM4_CH4, LL_DMAMUX_REQ1_3_TIM4_UP, LL_DMAMUX_REQ1_3_TIM6_UP, LL_DMAMUX_REQ1_3_USART1_TX, LL_DMAMUX_REQ1_3_USART2_TX, LL_DMAMUX_REQ1_3_USART3_TX, LL_DMAMUX_REQ1_3_DAC_CH2, LL_DMAMUX_REQ1_3_DAC_CH4, LL_DMAMUX_REQ1_3_ATK_B }
输出	无
返回值	无
资源使用	
说明	

2.14.45 LL_DMAMUX_GetRequestID

2.14.45.1 功能介绍

获取 DMAMUX 请求信号源标识。

2.14.45.2 接口定义

函数接口	uint32_t LL_DMAMUX_GetRequestID (DMAMUX_TypeDef *
------	--

	DMAMUXx, uint32_t Channel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1, LL_DMAMUX_CHANNEL_2, LL_DMAMUX_CHANNEL_3 }
输出	无
返回值	{ LL_DMAMUX_REQ_MEM2MEM, LL_DMAMUX_REQ_GENERATOR0, LL_DMAMUX_REQ_GENERATOR1, LL_DMAMUX_REQ_ADC1, In Channel 0 or 2: LL_DMAMUX_REQ0_2_I2C1_RX, LL_DMAMUX_REQ0_2_I2C2_RX, LL_DMAMUX_REQ0_2_LPUART1_RX, LL_DMAMUX_REQ0_2_SPI1_RX, LL_DMAMUX_REQ0_2_SPI2_RX, LL_DMAMUX_REQ0_2_TIM1_CH1, LL_DMAMUX_REQ0_2_TIM1_CH3, LL_DMAMUX_REQ0_2_TIM1_TRIG_COM, LL_DMAMUX_REQ0_2_TIM2_CH1, LL_DMAMUX_REQ0_2_TIM2_CH3, LL_DMAMUX_REQ0_2_TIM2_TRIG, LL_DMAMUX_REQ0_2_TIM3_CH1, LL_DMAMUX_REQ0_2_TIM3_CH3, LL_DMAMUX_REQ0_2_TIM3_TRIG, LL_DMAMUX_REQ0_2_TIM4_CH1, LL_DMAMUX_REQ0_2_TIM4_CH3, LL_DMAMUX_REQ0_2_TIM4_TRIG, LL_DMAMUX_REQ0_2_TIM5_UP, LL_DMAMUX_REQ0_2_TIM7_UP, LL_DMAMUX_REQ0_2_USART1_RX, LL_DMAMUX_REQ0_2_USART2_RX, LL_DMAMUX_REQ0_2_USART3_RX, LL_DMAMUX_REQ0_2_DAC_CH1, LL_DMAMUX_REQ0_2_DAC_CH3, LL_DMAMUX_REQ0_2_ATK_A, In Channel 1 or 3 : LL_DMAMUX_REQ1_3_I2C1_TX, LL_DMAMUX_REQ1_3_I2C2_TX, LL_DMAMUX_REQ1_3_LPUART1_TX, LL_DMAMUX_REQ1_3_SPI1_TX, LL_DMAMUX_REQ1_3_SPI2_TX, LL_DMAMUX_REQ1_3_TIM1_CH2, LL_DMAMUX_REQ1_3_TIM1_CH4, LL_DMAMUX_REQ1_3_TIM1_UP, LL_DMAMUX_REQ1_3_TIM2_CH2, LL_DMAMUX_REQ1_3_TIM2_CH4, LL_DMAMUX_REQ1_3_TIM2_UP, LL_DMAMUX_REQ1_3_TIM3_CH2, LL_DMAMUX_REQ1_3_TIM3_CH4, LL_DMAMUX_REQ1_3_TIM3_UP,

	LL_DMAMUX_REQ1_3_TIM4_CH2, LL_DMAMUX_REQ1_3_TIM4_CH4, LL_DMAMUX_REQ1_3_TIM4_UP, LL_DMAMUX_REQ1_3_TIM6_UP, LL_DMAMUX_REQ1_3_USART1_TX, LL_DMAMUX_REQ1_3_USART2_TX, LL_DMAMUX_REQ1_3_USART3_TX, LL_DMAMUX_REQ1_3_DAC_CH2, LL_DMAMUX_REQ1_3_DAC_CH4, LL_DMAMUX_REQ1_3_ATK_B }
资源使用	
说明	

2.14.46 LL_DMAMUX_DisableEventGeneration

2.14.46.1 功能介绍

禁止使能 DMAMUX 事件生成。

2.14.46.2 接口定义

函数接口	void LL_DMAMUX_DisableEventGeneration (DMAMUX_TypeDef * DMAMUXx, uint32_t Channel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1, LL_DMAMUX_CHANNEL_2, LL_DMAMUX_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.47 LL_DMAMUX_EnableEventGeneration

2.14.47.1 功能介绍

使能 DMAMUX 事件生成。

2.14.47.2 接口定义

函数接口	void LL_DMAMUX_EnableEventGeneration (DMAMUX_TypeDef * DMAMUXx, uint32_t Channel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1, LL_DMAMUX_CHANNEL_2, LL_DMAMUX_CHANNEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.14.48 LL_DMAMUX_IsEnabledEventGeneration

2.14.48.1 功能介绍

检查是否使能 DMAMUX 事件生成。

2.14.48.2 接口定义

函数接口	uint32_t LL_DMAMUX_IsEnabledEventGeneration (DMAMUX_TypeDef * DMAMUXx, uint32_t Channel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1, LL_DMAMUX_CHANNEL_2, LL_DMAMUX_CHANNEL_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.49 LL_DMAMUX_SetEventGenerationNumber

2.14.49.1 功能介绍

设置 DMAMUX 生成事件的 DMA 请求数。

2.14.49.2 接口定义

函数接口	void LL_DMAMUX_SetEventGenerationNumber (DMAMUX_TypeDef * DMAMUXx, uint32_t Channel, uint32_t RequestNb)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1 } uint32_t RequestNb: [1, 32]
输出	无
返回值	无
资源使用	
说明	

2.14.50 LL_DMAMUX_GetEventGenerationNumber

2.14.50.1 功能介绍

获取 DMAMUX 生成事件的 DMA 请求数。

2.14.50.2 接口定义

函数接口	uint32_t LL_DMAMUX_GetEventGenerationNumber (DMAMUX_TypeDef * DMAMUXx, uint32_t Channel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t Channel: { LL_DMAMUX_CHANNEL_0, LL_DMAMUX_CHANNEL_1 }
输出	无
返回值	[1, 32]

资源使用	
说明	

2.14.51 LL_DMAMUX_EnableRequestGen

2.14.51.1 功能介绍

使能请求生成器。

2.14.51.2 接口定义

函数接口	void LL_DMAMUX_EnableRequestGen (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	无
资源使用	
说明	

2.14.52 LL_DMAMUX_DisableRequestGen

2.14.52.1 功能介绍

禁止使能请求生成器。

2.14.52.2 接口定义

函数接口	void LL_DMAMUX_DisableRequestGen (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	无
资源使用	
说明	

2.14.53 LL_DMAMUX_IsEnabledRequestGen

2.14.53.1 功能介绍

检查是否使能请求生成器。

2.14.53.2 接口定义

函数接口	uint32_t LL_DMAMUX_IsEnabledRequestGen (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	{0,1}

资源使用	
说明	

2.14.54 LL_DMAMUX_SetRequestGenPolarity

2.14.54.1 功能介绍

设置 DMAMUX 触发极性。

2.14.54.2 接口定义

函数接口	void LL_DMAMUX_SetRequestGenPolarity (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel, uint32_t Polarity)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 } uint32_t Polarity: { LL_DMAMUX_REQ_GEN_NO_EVENT, LL_DMAMUX_REQ_GEN_POL_RISING , LL_DMAMUX_REQ_GEN_POL_FALLING, LL_DMAMUX_REQ_GEN_POL_RISING_FALLING }
输出	无
返回值	无
资源使用	
说明	

2.14.55 LL_DMAMUX_GetRequestGenPolarity

2.14.55.1 功能介绍

获取 DMAMUX 触发极性。

2.14.55.2 接口定义

函数接口	uint32_t LL_DMAMUX_GetRequestGenPolarity (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	{ LL_DMAMUX_REQ_GEN_NO_EVENT, LL_DMAMUX_REQ_GEN_POL_RISING , LL_DMAMUX_REQ_GEN_POL_FALLING, LL_DMAMUX_REQ_GEN_POL_RISING_FALLING }
资源使用	
说明	

2.14.56 LL_DMAMUX_SetGenRequestNb

2.14.56.1 功能介绍

设置 DMAMUX 生成 DMA 请求数。

2.14.56.2 接口定义

函数接口	void LL_DMAMUX_SetGenRequestNb (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel, uint32_t RequestNb)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 } uint32_t RequestNb: [1, 32]
输出	无
返回值	无
资源使用	
说明	

2.14.57 LL_DMAMUX_GetGenRequestNb
2.14.57.1 功能介绍

获取 DMAMUX 生成 DMA 请求数。

2.14.57.2 接口定义

函数接口	uint32_t LL_DMAMUX_GetGenRequestNb (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	[1, 32]
资源使用	
说明	

2.14.58 LL_DMAMUX_SetRequestSignalID
2.14.58.1 功能介绍

设置 DMAMUX 触发信号源标识。

2.14.58.2 接口定义

函数接口	void LL_DMAMUX_SetRequestSignalID (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel, uint32_t RequestSignalID)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 } uint32_t RequestSignalID: { LL_DMAMUX_REQ_GEN_EXTI_LINE1, LL_DMAMUX_REQ_GEN_EXTI_LINE2, LL_DMAMUX_REQ_GEN_EXTI_LINE3, LL_DMAMUX_REQ_GEN_DMAMUX_CH0, LL_DMAMUX_REQ_GEN_DMAMUX_CH1, LL_DMAMUX_REQ_GEN_LPTIM1_OUT,

	LL_DMAMUX_REQ_GEN_TIM4_OC2, LL_DMAMUX_REQ_GEN_LCD_TRIG }
输出	无
返回值	无
资源使用	
说明	

2.14.59 LL_DMAMUX_GetRequestSignalID

2.14.59.1 功能介绍

获取 DMAMUX 触发信号源标识。

2.14.59.2 接口定义

函数接口	uint32_t LL_DMAMUX_GetRequestSignalID (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	{ LL_DMAMUX_REQ_GEN_EXTI_LINE1, LL_DMAMUX_REQ_GEN_EXTI_LINE2, LL_DMAMUX_REQ_GEN_EXTI_LINE3, LL_DMAMUX_REQ_GEN_DMAMUX_CH0, LL_DMAMUX_REQ_GEN_DMAMUX_CH1, LL_DMAMUX_REQ_GEN_LPTIM1_OUT, LL_DMAMUX_REQ_GEN_TIM4_OC2, LL_DMAMUX_REQ_GEN_LCD_TRIG }
资源使用	
说明	

2.14.60 LL_DMAMUX_IsActiveFlag_RGO

2.14.60.1 功能介绍

获取请求生成器通道 1 触发溢出标志。

2.14.60.2 接口定义

函数接口	uint32_t LL_DMAMUX_IsActiveFlag_RGO (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.14.61 LL_DMAMUX_ClearFlag_RGO

2.14.61.1 功能介绍

清除请求生成器通道 1 触发溢出标志。

2.14.61.2 接口定义

函数接口	void LL_DMAMUX_ClearFlag_RGO (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	无
资源使用	
说明	

2.14.62 LL_DMAMUX_EnableIT_RGO

2.14.62.1 功能介绍

使能触发溢出中断。

2.14.62.2 接口定义

函数接口	void LL_DMAMUX_EnableIT_RGO (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	无
资源使用	
说明	

2.14.63 LL_DMAMUX_DisableIT_RGO

2.14.63.1 功能介绍

禁止使能触发溢出中断。

2.14.63.2 接口定义

函数接口	void LL_DMAMUX_DisableIT_RGO (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	无
资源使用	
说明	

2.14.64 LL_DMAMUX_IsEnabledIT_RGO

2.14.64.1 功能介绍

检查是否使能触发溢出中断。

2.14.64.2 接口定义

函数接口	uint32_t LL_DMAMUX_IsEnabledIT_RGO (DMAMUX_TypeDef * DMAMUXx, uint32_t RequestGenChannel)
输入	DMAMUX_TypeDef * DMAMUXx uint32_t RequestGenChannel: { LL_DMAMUX_REQ_GEN_0, LL_DMAMUX_REQ_GEN_1 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.15 HIDV 模块

2.15.1 LL_HDIV_GetDividend

2.15.1.1 功能介绍

获取 HDIV 被除数。

2.15.1.2 接口定义

函数接口	uint32_t LL_HDIV_GetDividend (void)
输入	无
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.15.2 LL_HDIV_SetDividend

2.15.2.1 功能介绍

设置 HDIV 被除数。

2.15.2.2 接口定义

函数接口	void LL_HDIV_SetDividend (uint32_t DividendValue)
输入	uint32_t DividendValue: [0x00000000,0x FFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.15.3 LL_HDIV_GetDivisor

2.15.3.1 功能介绍

获取 HDIV 除数。

2.15.3.2 接口定义

函数接口	uint32_t LL_HDIV_GetDivisor (void)
输入	无
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.15.4 LL_HDIV_SetDivisor

2.15.4.1 功能介绍

设置 HDIV 除数。

2.15.4.2 接口定义

函数接口	void LL_HDIV_SetDivisor (uint32_t DivisorValue)
输入	uint32_t DivisorValue: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.15.5 LL_HDIV_GetSign

2.15.5.1 功能介绍

获取 HDIV 符号。

2.15.5.2 接口定义

函数接口	uint32_t LL_HDIV_GetSign (void)
输入	无
输出	无
返回值	{ LL_HDIV_SIGN_OFF, LL_HDIV_SIGN_ON }
资源使用	
说明	

2.15.6 LL_HDIV_SetSign

2.15.6.1 功能介绍

设置 HDIV 符号。

2.15.6.2 接口定义

函数接口	void LL_HDIV_SetSign (uint32_t SignValue)
输入	uint32_t SignValue: { LL_HDIV_SIGN_OFF, LL_HDIV_SIGN_ON }
输出	无
返回值	无
资源使用	

说明	
----	--

2.15.7 LL_HDIV_IsActiveFlag_DivisorZero

2.15.7.1 功能介绍

检查除数为零警告标志。

2.15.7.2 接口定义

函数接口	uint32_t LL_HDIV_IsActiveFlag_DivisorZero (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.15.8 LL_HDIV_IsActiveFlag_Finished

2.15.8.1 功能介绍

检查除法运算结束标志。

2.15.8.2 接口定义

函数接口	uint32_t LL_HDIV_IsActiveFlag_Finished (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.15.9 LL_HDIV_GetQuotient

2.15.9.1 功能介绍

获取 HDIV 商。

2.15.9.2 接口定义

函数接口	uint32_t LL_HDIV_GetQuotient (void)
输入	无
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.15.10 LL_HDIV_GetRemainder

2.15.10.1 功能介绍

获取 HDIV 余数。

2.15.10.2 接口定义

函数接口	uint32_t LL_HDIV_GetRemainder (void)
输入	无

输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.16 I2C 模块

属性	类型	字段名	含义
I2C_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	OAR1	地址寄存器 1
读写	uint32_t	OAR2	地址寄存器 2
读写	uint32_t	TIMINGR	时钟配置寄存器
读写	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断清除寄存器
只读	uint32_t	RXDR	接收数据寄存器
读写	uint32_t	TXDR	发送数据寄存器
LL_I2C_InitTypeDef			
读写	uint32_t	Timing	时钟配置
读写	uint32_t	DigitalFilter	数字噪声滤波
读写	uint32_t	OwnAddress1	设备从模式地址 1
读写	uint32_t	TypeAcknowledge	应答信号类型
读写	uint32_t	OwnAddrSize	设备从模式地址 1 大小

2.16.1 LL_I2C_DMA_GetRegTxAddr

2.16.1.1 功能介绍

获取用于 DMA 传输的 I2C 寄存器地址。

2.16.1.2 接口定义

函数接口	uint32_t LL_I2C_DMA_GetRegTxAddr (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	寄存器地址
资源使用	
说明	

2.16.2 LL_I2C_DMA_GetRegRxAddr

2.16.2.1 功能介绍

获取用于 DMA 接收的 I2C 寄存器地址。

2.16.2.2 接口定义

函数接口	uint32_t LL_I2C_DMA_GetRegRxAddr (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	寄存器地址
资源使用	
说明	

2.16.3 LL_I2C_Enable

2.16.3.1 功能介绍

使能 I2C。

2.16.3.2 接口定义

函数接口	void LL_I2C_Enable (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.4 LL_I2C_Disable

2.16.4.1 功能介绍

禁止使能 I2C。

2.16.4.2 接口定义

函数接口	void LL_I2C_Disable (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.5 LL_I2C_IsEnabled

2.16.5.1 功能介绍

检查是否使能 I2C。

2.16.5.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabled (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	{0,1}
资源使用	
说明	

2.16.6 LL_I2C_SetDigitalFilter

2.16.6.1 功能介绍

配置数字噪声滤波器。

2.16.6.2 接口定义

函数接口	void LL_I2C_SetDigitalFilter (I2C_TypeDef * I2Cx, uint32_t DigitalFilter)
输入	I2C_TypeDef * I2Cx uint32_t DigitalFilter: [0x00, 0x0F]
输出	无
返回值	无
资源使用	
说明	

2.16.7 LL_I2C_GetDigitalFilter

2.16.7.1 功能介绍

获取数字噪声滤波器。

2.16.7.2 接口定义

函数接口	uint32_t LL_I2C_GetDigitalFilter (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0x0F]
资源使用	
说明	

2.16.8 LL_I2C_EnableDMAReq_TX

2.16.8.1 功能介绍

使能 DMA 传输请求。

2.16.8.2 接口定义

函数接口	void LL_I2C_EnableDMAReq_TX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.9 LL_I2C_DisableDMAReq_TX

2.16.9.1 功能介绍

禁止使能 DMA 传输请求。

2.16.9.2 接口定义

函数接口	void LL_I2C_DisableDMAReq_TX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.10 LL_I2C_IsEnabledDMAReq_TX

2.16.10.1 功能介绍

检查是否使能 DMA 传输请求。

2.16.10.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledDMAReq_TX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.11 LL_I2C_EnableDMAReq_RX

2.16.11.1 功能介绍

使能 DMA 传输接收请求。

2.16.11.2 接口定义

函数接口	void LL_I2C_EnableDMAReq_RX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.12 LL_I2C_DisableDMAReq_RX

2.16.12.1 功能介绍

禁止使能 DMA 传输接收请求。

2.16.12.2 接口定义

函数接口	void LL_I2C_DisableDMAReq_RX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	无
资源使用	
说明	

2.16.13 LL_I2C_IsEnabledDMAReq_RX

2.16.13.1 功能介绍

检查是否使能 DMA 传输接收请求。

2.16.13.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledDMAReq_RX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.14 LL_I2C_DMA_GetRegAddr

2.16.14.1 功能介绍

获取用于 DMA 传输的数据寄存器地址。

2.16.14.2 接口定义

函数接口	uint32_t LL_I2C_DMA_GetRegAddr (I2C_TypeDef * I2Cx, uint32_t Direction)
输入	I2C_TypeDef * I2Cx uint32_t Direction: { LL_I2C_DMA_REG_DATA_TRANSMIT, LL_I2C_DMA_REG_DATA_RECEIVE }
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.16.15 LL_I2C_EnableClockStretching

2.16.15.1 功能介绍

使能时钟低电平延长。

2.16.15.2 接口定义

函数接口	void LL_I2C_EnableClockStretching (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.16 LL_I2C_DisableClockStretching

2.16.16.1 功能介绍

禁止使能时钟低电平延长。

2.16.16.2 接口定义

函数接口	void LL_I2C_DisableClockStretching (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.17 LL_I2C_IsEnabledClockStretching

2.16.17.1 功能介绍

检查是否使能时钟低电平延长。

2.16.17.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledClockStretching (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.18 LL_I2C_EnableSlaveByteControl

2.16.18.1 功能介绍

使能从模式字节计数控制。

2.16.18.2 接口定义

函数接口	void LL_I2C_EnableSlaveByteControl (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.19 LL_I2C_DisableSlaveByteControl

2.16.19.1 功能介绍

禁止使能从模式字节计数控制。

2.16.19.2 接口定义

函数接口	void LL_I2C_DisableSlaveByteControl (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	无
资源使用	
说明	

2.16.20 LL_I2C_IsEnabledSlaveByteControl

2.16.20.1 功能介绍

检查是否使能从模式字节计数控制。

2.16.20.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledSlaveByteControl (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.21 LL_I2C_EnableWakeUpFromStop

2.16.21.1 功能介绍

使能从 Stop 模式唤醒。

2.16.21.2 接口定义

函数接口	void LL_I2C_EnableWakeUpFromStop (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.22 LL_I2C_DisableWakeUpFromStop

2.16.22.1 功能介绍

禁止使能从 Stop 模式唤醒。

2.16.22.2 接口定义

函数接口	void LL_I2C_DisableWakeUpFromStop (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.23 LL_I2C_IsEnabledWakeUpFromStop

2.16.23.1 功能介绍

检查是否使能从 Stop 模式唤醒。

2.16.23.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledWakeUpFromStop (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.24 LL_I2C_EnableGeneralCall

2.16.24.1 功能介绍

使能广播地址。

2.16.24.2 接口定义

函数接口	void LL_I2C_EnableGeneralCall (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.25 LL_I2C_DisableGeneralCall

2.16.25.1 功能介绍

禁止使能广播地址。

2.16.25.2 接口定义

函数接口	void LL_I2C_DisableGeneralCall (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.26 LL_I2C_IsEnabledGeneralCall

2.16.26.1 功能介绍

检查是否使能广播地址。

2.16.26.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledGeneralCall (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.27 LL_I2C_SetMasterAddressingMode

2.16.27.1 功能介绍

设置主模式 7 位或 10 位寻址模式。

2.16.27.2 接口定义

函数接口	void LL_I2C_SetMasterAddressingMode (I2C_TypeDef * I2Cx, uint32_t AddressingMode)
输入	I2C_TypeDef * I2Cx uint32_t AddressingMode: { LL_I2C_ADDRESSING_MODE_7BIT, LL_I2C_ADDRESSING_MODE_10BIT }
输出	无
返回值	无
资源使用	
说明	

2.16.28 LL_I2C_GetMasterAddressingMode

2.16.28.1 功能介绍

获取主模式 7 位或 10 位寻址模式。

2.16.28.2 接口定义

函数接口	uint32_t LL_I2C_GetMasterAddressingMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{ LL_I2C_ADDRESSING_MODE_7BIT, LL_I2C_ADDRESSING_MODE_10BIT }
资源使用	
说明	

2.16.29 LL_I2C_SetOwnAddress1

2.16.29.1 功能介绍

设置从模式地址 1。

2.16.29.2 接口定义

函数接口	void LL_I2C_SetOwnAddress1 (I2C_TypeDef * I2Cx, uint32_t OwnAddress1, uint32_t OwnAddrSize)
输入	I2C_TypeDef * I2Cx uint32_t OwnAddress1: [0x000, 0x3FF] uint32_t OwnAddrSize: { LL_I2C_OWNAADDRESS1_7BIT, LL_I2C_OWNAADDRESS1_10BIT }
输出	无
返回值	无
资源使用	
说明	

2.16.30 LL_I2C_GetOwnAddress1

2.16.30.1 功能介绍

获取从模式地址 1。

2.16.30.2 接口定义

函数接口	uint32_t LL_I2C_GetOwnAddress1 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x000, 0x3FF]
资源使用	
说明	

2.16.31 LL_I2C_EnableOwnAddress1

2.16.31.1 功能介绍

使能从模式地址 1。

2.16.31.2 接口定义

函数接口	void LL_I2C_EnableOwnAddress1 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.32 LL_I2C_DisableOwnAddress1

2.16.32.1 功能介绍

禁止使能从模式地址 1。

2.16.32.2 接口定义

函数接口	void LL_I2C_DisableOwnAddress1 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.33 LL_I2C_IsEnabledOwnAddress1

2.16.33.1 功能介绍

检查是否使能从模式地址 1。

2.16.33.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledOwnAddress1 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	{0,1}
资源使用	
说明	

2.16.34 LL_I2C_SetOwnAddress2

2.16.34.1 功能介绍

设置 7 位从模式地址 2。

2.16.34.2 接口定义

函数接口	void LL_I2C_SetOwnAddress2 (I2C_TypeDef * I2Cx, uint32_t OwnAddress2, uint32_t OwnAddrMask)
输入	I2C_TypeDef * I2Cx uint32_t OwnAddress2 [0x00, 0x7F] uint32_t OwnAddrMask {LL_I2C_OWNADDRESS2_NOMASK, LL_I2C_OWNADDRESS2_MASK01, LL_I2C_OWNADDRESS2_MASK02, LL_I2C_OWNADDRESS2_MASK03, LL_I2C_OWNADDRESS2_MASK04, LL_I2C_OWNADDRESS2_MASK05, LL_I2C_OWNADDRESS2_MASK06, LL_I2C_OWNADDRESS2_MASK07 }
输出	无
返回值	无
资源使用	
说明	

2.16.35 LL_I2C_GetOwnAddress2

2.16.35.1 功能介绍

获取 7 位从模式地址 2。

2.16.35.2 接口定义

函数接口	uint32_t LL_I2C_GetOwnAddress2 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0x7F]
资源使用	
说明	

2.16.36 LL_I2C_EnableOwnAddress2

2.16.36.1 功能介绍

使能从模式地址 2。

2.16.36.2 接口定义

函数接口	void LL_I2C_EnableOwnAddress2 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.37 LL_I2C_DisableOwnAddress2

2.16.37.1 功能介绍

禁止使能从模式地址 2。

2.16.37.2 接口定义

函数接口	void LL_I2C_DisableOwnAddress2 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.38 LL_I2C_IsEnabledOwnAddress2

2.16.38.1 功能介绍

检查是否使能从模式地址 2。

2.16.38.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledOwnAddress2 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.39 LL_I2C_SetTiming

2.16.39.1 功能介绍

设置 I2C 时钟配置。

2.16.39.2 接口定义

函数接口	void LL_I2C_SetTiming (I2C_TypeDef * I2Cx, uint32_t Timing)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.16.40 LL_I2C_GetTimingPrescaler

2.16.40.1 功能介绍

获取 I2C 时钟预分频。

2.16.40.2 接口定义

函数接口	uint32_t LL_I2C_GetTimingPrescaler (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x0, 0xF]
资源使用	
说明	

2.16.41 LL_I2C_SetTimingPrescaler

2.16.41.1 功能介绍

设置 I2C 时钟预分频。

2.16.41.2 接口定义

函数接口	void LL_I2C_SetTimingPrescale (I2C_TypeDef * I2Cx, uint32_t Psc)
输入	I2C_TypeDef * I2Cx uint32_t Psc: [0x0, 0xF]
输出	无
返回值	无
资源使用	
说明	

2.16.42 LL_I2C_GetClockLowPeriod

2.16.42.1 功能介绍

获取 I2C SCL 低电平时间。

2.16.42.2 接口定义

函数接口	uint32_t LL_I2C_GetClockLowPeriod (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.16.43 LL_I2C_SetClockLowPeriod

2.16.43.1 功能介绍

设置 I2C SCL 低电平时间。

2.16.43.2 接口定义

函数接口	void LL_I2C_SetClockLowPeriod (I2C_TypeDef * I2Cx, uint32_t SCLLowPeriod)
------	---

输入	I2C_TypeDef * I2Cx uint32_t SCLLowPeriod: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.16.44 LL_I2C_GetClockHighPeriod

2.16.44.1 功能介绍

获取 I2C SCL 高电平时间。

2.16.44.2 接口定义

函数接口	uint32_t LL_I2C_GetClockHighPeriod (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.16.45 LL_I2C_SetClockHighPeriod

2.16.45.1 功能介绍

设置 I2C SCL 高电平时间。

2.16.45.2 接口定义

函数接口	void LL_I2C_SetClockHighPeriod (I2C_TypeDef * I2Cx, uint32_t SCLHighPeriod)
输入	I2C_TypeDef * I2Cx uint32_t SCLHighPeriod: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.16.46 LL_I2C_GetDataHoldTime

2.16.46.1 功能介绍

获取 I2C 数据保持时间。

2.16.46.2 接口定义

函数接口	uint32_t LL_I2C_GetDataHoldTime (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x0, 0xF]

资源使用	
说明	

2.16.47 LL_I2C_SetDataHoldTime

2.16.47.1 功能介绍

设置 I2C 数据保持时间。

2.16.47.2 接口定义

函数接口	void LL_I2C_SetDataHoldTime (I2C_TypeDef * I2Cx, uint32_t SDAHoldTime)
输入	I2C_TypeDef * I2Cx uint32_t SDAHoldTime: [0x0, 0xF]
输出	无
返回值	无
资源使用	
说明	

2.16.48 LL_I2C_GetDataSetupTime

2.16.48.1 功能介绍

获取 I2C 数据建立时间。

2.16.48.2 接口定义

函数接口	uint32_t LL_I2C_GetDataSetupTime (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x0, 0xF]
资源使用	
说明	

2.16.49 LL_I2C_SetDataSetupTime

2.16.49.1 功能介绍

设置 I2C 数据建立时间。

2.16.49.2 接口定义

函数接口	void LL_I2C_SetDataSetupTime (I2C_TypeDef * I2Cx, uint32_t SDAStepTime)
输入	I2C_TypeDef * I2Cx uint32_t SDAStepTime: [0x0, 0xF]
输出	无
返回值	无
资源使用	
说明	

2.16.50 LL_I2C_EnableIT_TX

2.16.50.1 功能介绍

使能 I2C 发送中断。

2.16.50.2 接口定义

函数接口	void LL_I2C_EnableIT_TX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.51 LL_I2C_DisableIT_TX

2.16.51.1 功能介绍

禁止使能 I2C 发送中断。

2.16.51.2 接口定义

函数接口	void LL_I2C_DisableIT_TX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.52 LL_I2C_IsEnabledIT_TX

2.16.52.1 功能介绍

检查是否使能 I2C 发送中断。

2.16.52.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_TX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.53 LL_I2C_EnableIT_RX

2.16.53.1 功能介绍

使能 I2C 接收中断。

2.16.53.2 接口定义

函数接口	void LL_I2C_EnableIT_RX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	无
资源使用	
说明	

2.16.54 LL_I2C_DisableIT_RX

2.16.54.1 功能介绍

禁止使能 I2C 接收中断。

2.16.54.2 接口定义

函数接口	void LL_I2C_DisableIT_RX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.55 LL_I2C_IsEnabledIT_RX

2.16.55.1 功能介绍

检查是否使能 I2C 接收中断。

2.16.55.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_RX (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.56 LL_I2C_EnableIT_ADDR

2.16.56.1 功能介绍

使能 I2C 地址匹配中断。

2.16.56.2 接口定义

函数接口	void LL_I2C_EnableIT_ADDR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.57 LL_I2C_DisableIT_ADDR

2.16.57.1 功能介绍

禁止使能地址匹配中断。

2.16.57.2 接口定义

函数接口	void LL_I2C_DisableIT_ADDR(I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.58 LL_I2C_IsEnabledIT_ADDR

2.16.58.1 功能介绍

检查是否使能地址匹配中断。

2.16.58.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_ADDR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.59 LL_I2C_EnableIT_NACK

2.16.59.1 功能介绍

使能 I2C NACK 应答中断。

2.16.59.2 接口定义

函数接口	void LL_I2C_EnableIT_NACK (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.60 LL_I2C_DisableIT_NACK

2.16.60.1 功能介绍

禁止使能 I2C NACK 应答中断。

2.16.60.2 接口定义

函数接口	void LL_I2C_DisableIT_NACK (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.61 LL_I2C_IsEnabledIT_NACK

2.16.61.1 功能介绍

检查是否使能 NACK 应答中断。

2.16.61.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_NACK (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.62 LL_I2C_EnableIT_STOP

2.16.62.1 功能介绍

使能 I2C 停止位检测中断。

2.16.62.2 接口定义

函数接口	void LL_I2C_EnableIT_STOP (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.63 LL_I2C_DisableIT_STOP

2.16.63.1 功能介绍

禁止使能 I2C 停止位检测中断。

2.16.63.2 接口定义

函数接口	void LL_I2C_DisableIT_STOP (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.64 LL_I2C_IsEnabledIT_STOP

2.16.64.1 功能介绍

检查是否使能停止位检测中断。

2.16.64.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_STOP (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	{0,1}
资源使用	
说明	

2.16.65 LL_I2C_EnableIT_TC

2.16.65.1 功能介绍

使能 I2C 传输完成中断。

2.16.65.2 接口定义

函数接口	void LL_I2C_EnableIT_TC (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.66 LL_I2C_DisableIT_TC

2.16.66.1 功能介绍

禁止使能 I2C 传输完成中断。

2.16.66.2 接口定义

函数接口	void LL_I2C_DisableIT_TC (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.67 LL_I2C_IsEnabledIT_TC

2.16.67.1 功能介绍

检查是否使能传输完成中断。

2.16.67.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_TC (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.68 LL_I2C_EnableIT_ERR

2.16.68.1 功能介绍

使能 I2C 错误检测中断。

2.16.68.2 接口定义

函数接口	void LL_I2C_EnableIT_ERR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.69 LL_I2C_DisableIT_ERR

2.16.69.1 功能介绍

禁止使能 I2C 错误检测中断。

2.16.69.2 接口定义

函数接口	void LL_I2C_DisableIT_ERR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.70 LL_I2C_IsEnabledIT_ERR

2.16.70.1 功能介绍

检查是否使能错误检测中断。

2.16.70.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT_ERR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.71 LL_I2C_EnableIT

2.16.71.1 功能介绍

使能 I2C。

2.16.71.2 接口定义

函数接口	void LL_I2C_EnableIT (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.72 LL_I2C_DisableIT

2.16.72.1 功能介绍

禁止使能 I2C。

2.16.72.2 接口定义

函数接口	void LL_I2C_DisableIT (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.73 LL_I2C_IsEnabledIT

2.16.73.1 功能介绍

检查是否使能 I2C。

2.16.73.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledIT (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.74 LL_I2C_IsActiveFlag_TXE

2.16.74.1 功能介绍

检查是否发送数据寄存器为空。

2.16.74.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_TXE (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.75 LL_I2C_IsActiveFlag_TXIS

2.16.75.1 功能介绍

检查发送中断状态。

2.16.75.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_TXIS (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	{0,1}
资源使用	
说明	

2.16.76 LL_I2C_IsActiveFlag_RXNE

2.16.76.1 功能介绍

检查是否接收数据寄存器非空。

2.16.76.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_RXNE (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.77 LL_I2C_IsActiveFlag_ADDR

2.16.77.1 功能介绍

检查从模式地址匹配标志。

2.16.77.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_ADDR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.78 LL_I2C_IsActiveFlag_NACK

2.16.78.1 功能介绍

检查接收到 NACK 标志。

2.16.78.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_NACK (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.79 LL_I2C_IsActiveFlag_STOP

2.16.79.1 功能介绍

检查停止位标志。

2.16.79.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_STOP (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.80 LL_I2C_IsActiveFlag_TC

2.16.80.1 功能介绍

检查是否主模式传输完成。

2.16.80.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_TC (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.81 LL_I2C_IsActiveFlag_TCR

2.16.81.1 功能介绍

检查是否传输完成，等待重新写入 NBYTES。

2.16.81.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_TCR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.82 LL_I2C_IsActiveFlag_BERR

2.16.82.1 功能介绍

检查总线错误标志。

2.16.82.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_BERR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.83 LL_I2C_IsActiveFlag_ARLO

2.16.83.1 功能介绍

检查仲裁丢失标志。

2.16.83.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_ARLO (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.84 LL_I2C_IsActiveFlag_OVR

2.16.84.1 功能介绍

检查从模式溢出错误标志。

2.16.84.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_OVR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.85 LL_I2C_IsActiveFlag_TIMEOUT

2.16.85.1 功能介绍

检查超时错误标志。

2.16.85.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_TIMEOUT (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.86 LL_I2C_IsActiveFlag_BUSY

2.16.86.1 功能介绍

检查是否总线忙。

2.16.86.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag_BUSY (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	{0,1}
资源使用	
说明	

2.16.87 LL_I2C_IsActiveFlag

2.16.87.1 功能介绍

检查标志位状态。

2.16.87.2 接口定义

函数接口	uint32_t LL_I2C_IsActiveFlag (I2C_TypeDef * I2Cx, uint32_t Flag)
输入	I2C_TypeDef * I2Cx uint32_t Flag: { LL_I2C_ISR_TXE, LL_I2C_ISR_TXIS, LL_I2C_ISR_RXNE, LL_I2C_ISR_ADDR, LL_I2C_ISR_NACKF, LL_I2C_ISR_STOPF, LL_I2C_ISR_TC, LL_I2C_ISR_TCR, LL_I2C_ISR_BERR, LL_I2C_ISR_ARLO, LL_I2C_ISR_OVR, LL_I2C_ISR_BUSY, LL_I2C_ISR_TIMEOUT, LL_I2C_ISR_DIR }
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.88 LL_I2C_ClearFlag_ADDR

2.16.88.1 功能介绍

清除从模式地址匹配标志。

2.16.88.2 接口定义

函数接口	void LL_I2C_ClearFlag_ADDR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.89 LL_I2C_ClearFlag_TXIS

2.16.89.1 功能介绍

清除发送中断。

2.16.89.2 接口定义

函数接口	void LL_I2C_ClearFlag_TXIS (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	

说明	
----	--

2.16.90 LL_I2C_ClearFlag_NACK

2.16.90.1 功能介绍

清除从 NACK 位标志。

2.16.90.2 接口定义

函数接口	void LL_I2C_ClearFlag_NACK (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.91 LL_I2C_ClearFlag_STOP

2.16.91.1 功能介绍

清除停止位标志。

2.16.91.2 接口定义

函数接口	void LL_I2C_ClearFlag_STOP (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.92 LL_I2C_ClearFlag_TXE

2.16.92.1 功能介绍

发送数据寄存器为空。

2.16.92.2 接口定义

函数接口	void LL_I2C_ClearFlag_TXE (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.93 LL_I2C_ClearFlag_BERR

2.16.93.1 功能介绍

清除总线错误标志。

2.16.93.2 接口定义

函数接口	void LL_I2C_ClearFlag_BERR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx

输出	无
返回值	无
资源使用	
说明	

2.16.94 LL_I2C_ClearFlag_ARLO

2.16.94.1 功能介绍

清除仲裁丢失标志。

2.16.94.2 接口定义

函数接口	void LL_I2C_ClearFlag_ARLO (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.95 LL_I2C_ClearFlag_OVR

2.16.95.1 功能介绍

清除溢出标志。

2.16.95.2 接口定义

函数接口	void LL_I2C_ClearFlag_OVR (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.96 LL_I2C_ClearFlag

2.16.96.1 功能介绍

清除标志。

2.16.96.2 接口定义

函数接口	void LL_I2C_ClearFlag (I2C_TypeDef * I2Cx, uint32_t Flag)
输入	I2C_TypeDef * I2Cx uint32_t Flag: { LL_I2C_ISR_ADDR, LL_I2C_ISR_NACKF, LL_I2C_ISR_STOPF, LL_I2C_ISR_BERR, LL_I2C_ISR_ARLO, LL_I2C_ISR_OVR }
输出	无
返回值	无
资源使用	
说明	

2.16.97 LL_I2C_EnableAutoEndMode

2.16.97.1 功能介绍

使能主模式自动结束模式。

2.16.97.2 接口定义

函数接口	void LL_I2C_EnableAutoEndMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.98 LL_I2C_DisableAutoEndMode

2.16.98.1 功能介绍

禁止使能主模式自动结束模式。

2.16.98.2 接口定义

函数接口	void LL_I2C_DisableAutoEndMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.99 LL_I2C_IsEnabledAutoEndMode

2.16.99.1 功能介绍

检查是否使能主模式自动结束模式。

2.16.99.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledAutoEndMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.100 LL_I2C_EnableReloadMode

2.16.100.1 功能介绍

使能重载模式。

2.16.100.2 接口定义

函数接口	void LL_I2C_EnableReloadMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	无
资源使用	
说明	

2.16.101 LL_I2C_DisableReloadMode

2.16.101.1 功能介绍

禁止使能重载模式。

2.16.101.2 接口定义

函数接口	void LL_I2C_DisableReloadMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.102 LL_I2C_IsEnabledReloadMode

2.16.102.1 功能介绍

检查是否使能重载模式。

2.16.102.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledReloadMode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.103 LL_I2C_SetTransferSize

2.16.103.1 功能介绍

设置待发送或接收的字节数。

2.16.103.2 接口定义

函数接口	void LL_I2C_SetTransferSize (I2C_TypeDef * I2Cx, uint32_t TransferSize)
输入	I2C_TypeDef * I2Cx uint32_t TransferSize: {0x00, 0xFF}
输出	无
返回值	无
资源使用	
说明	

2.16.104 LL_I2C_GetTransferSize

2.16.104.1 功能介绍

获取待发送或接收的字节数。

2.16.104.2 接口定义

函数接口	uint32_t LL_I2C_GetTransferSize (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0x00, 0xFF}
资源使用	
说明	

2.16.105 LL_I2C_AcknowledgeNextData

2.16.105.1 功能介绍

在地址接收匹配码或下一个接收字节之后，准备生成一个 ACK 或 NACK。

2.16.105.2 接口定义

函数接口	void LL_I2C_AcknowledgeNextData (I2C_TypeDef * I2Cx, uint32_t TypeAcknowledge)
输入	I2C_TypeDef * I2Cx uint32_t TypeAcknowledge: {LL_I2C_ACK, LL_I2C_NACK}
输出	无
返回值	无
资源使用	
说明	

2.16.106 LL_I2C_GenerateStartCondition

2.16.106.1 功能介绍

主模式起始位生成。

2.16.106.2 接口定义

函数接口	void LL_I2C_GenerateStartCondition (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.107 LL_I2C_IsGenerateStartCondition

2.16.107.1 功能介绍

检查主模式起始位是否生成。

2.16.107.2 接口定义

函数接口	uint32_t LL_I2C_IsGenerateStartCondition (I2C_TypeDef * I2Cx)
------	---

输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.108 LL_I2C_GenerateStopCondition

2.16.108.1 功能介绍

主模式停止位生成。

2.16.108.2 接口定义

函数接口	void LL_I2C_GenerateStopCondition (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.109 LL_I2C_IsGenerateStopCondition

2.16.109.1 功能介绍

检查主模式停止位是否生成。

2.16.109.2 接口定义

函数接口	uint32_t LL_I2C_IsGenerateStopCondition (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.110 LL_I2C_EnableAuto10BitRead

2.16.110.1 功能介绍

设置主设备读数据时只发送 10 位地址的前 7 位和读命令。

2.16.110.2 接口定义

函数接口	void LL_I2C_EnableAuto10BitRead (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.111 LL_I2C_DisableAuto10BitRead

2.16.111.1 功能介绍

设置主设备发送完整的 10 位地址读序列。

2.16.111.2 接口定义

函数接口	void LL_I2C_DisableAuto10BitRead (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	无
资源使用	
说明	

2.16.112 LL_I2C_IsEnabledAuto10BitRead

2.16.112.1 功能介绍

检查主模式发送接收 10 位地址头发送控制。

2.16.112.2 接口定义

函数接口	uint32_t LL_I2C_IsEnabledAuto10BitRead (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{0,1}
资源使用	
说明	

2.16.113 LL_I2C_SetTransferRequest

2.16.113.1 功能介绍

设置主模式数据传输方向。

2.16.113.2 接口定义

函数接口	void LL_I2C_SetTransferRequest (I2C_TypeDef * I2Cx, uint32_t TransferRequest)
输入	I2C_TypeDef * I2Cx uint32_t TransferRequest: { LL_I2C_REQUEST_WRITE, LL_I2C_REQUEST_READ }
输出	无
返回值	无
资源使用	
说明	

2.16.114 LL_I2C_GetTransferRequest

2.16.114.1 功能介绍

获取主模式数据传输方向。

2.16.114.2 接口定义

函数接口	uint32_t LL_I2C_GetTransferRequest (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{ LL_I2C_REQUEST_WRITE, LL_I2C_REQUEST_READ }
资源使用	
说明	

2.16.115 LL_I2C_SetSlaveAddr

2.16.115.1 功能介绍

设置主模式待发送的从设备地址。

2.16.115.2 接口定义

函数接口	void LL_I2C_SetSlaveAddr (I2C_TypeDef * I2Cx, uint32_t SlaveAddr)
输入	I2C_TypeDef * I2Cx uint32_t SlaveAddr: [0x00, 0x3F]
输出	无
返回值	无
资源使用	
说明	

2.16.116 LL_I2C_GetSlaveAddr

2.16.116.1 功能介绍

获取主模式待发送的从设备地址。

2.16.116.2 接口定义

函数接口	uint32_t LL_I2C_GetSlaveAddr (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0x3F]
资源使用	
说明	

2.16.117 LL_I2C_HandleTransfer

2.16.117.1 功能介绍

在开始传输或传输过程中处理 I2Cx 通信。

2.16.117.2 接口定义

函数接口	void LL_I2C_HandleTransfer (I2C_TypeDef * I2Cx, uint32_t SlaveAddr, uint32_t TransferSize, uint32_t EndMode, uint32_t Request)
输入	I2C_TypeDef * I2Cx uint32_t SlaveAddr uint32_t TransferSize:

	[0, 255] uint32_t EndMode: { LL_I2C_MODE_RELOAD, LL_I2C_MODE_AUTOEND, LL_I2C_MODE_SOFTEND } uint32_t Request: { LL_I2C_GENERATE_NOSTARTSTOP, LL_I2C_GENERATE_STOP, LL_I2C_GENERATE_START_READ, LL_I2C_GENERATE_START_WRITE, LL_I2C_GENERATE_RESTART_7BIT_READ, LL_I2C_GENERATE_RESTART_7BIT_WRITE, LL_I2C_GENERATE_RESTART_10BIT_READ, LL_I2C_GENERATE_RESTART_10BIT_WRITE }
输出	无
返回值	无
资源使用	
说明	

2.16.118 LL_I2C_GetTransferDirection

2.16.118.1 功能介绍

获取传输方向。

2.16.118.2 接口定义

函数接口	uint32_t LL_I2C_GetTransferDirection (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	{ LL_I2C_DIRECTION_WRITE, LL_I2C_DIRECTION_READ }
资源使用	
说明	

2.16.119 LL_I2C_GetAddressMatchCode

2.16.119.1 功能介绍

获取从模式匹配地址。

2.16.119.2 接口定义

函数接口	uint32_t LL_I2C_GetAddressMatchCode (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0x7F]
资源使用	
说明	

2.16.120 LL_I2C_ReceiveData8

2.16.120.1 功能介绍

读取接收数据寄存器。

2.16.120.2 接口定义

函数接口	uint8_t LL_I2C_ReceiveData8 (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.16.121 LL_I2C_TransmitData8

2.16.121.1 功能介绍

写入发送数据寄存器。

2.16.121.2 接口定义

函数接口	void LL_I2C_TransmitData8 (I2C_TypeDef * I2Cx, uint8_t Data)
输入	I2C_TypeDef * I2Cx uint8_t Data: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.16.122 LL_I2C_Init

2.16.122.1 功能介绍

I2C 初始化。

2.16.122.2 接口定义

函数接口	ErrorStatus LL_I2C_Init (I2C_TypeDef * I2Cx, LL_I2C_InitTypeDef * I2C_InitStruct)
输入	I2C_TypeDef * I2Cx LL_I2C_InitTypeDef * I2C_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.16.123 LL_I2C_DeInit

2.16.123.1 功能介绍

清除 I2C 初始化参数。

2.16.123.2 接口定义

函数接口	ErrorStatus LL_I2C_DeInit (I2C_TypeDef * I2Cx)
输入	I2C_TypeDef * I2Cx
输出	无

返回值	ErrorStatus
资源使用	
说明	

2.16.124 LL_I2C_StructInit

2.16.124.1 功能介绍

I2C 结构体初始化。

2.16.124.2 接口定义

函数接口	void LL_I2C_StructInit (LL_I2C_InitTypeDef * I2C_InitStruct)
输入	LL_I2C_InitTypeDef * I2C_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.17 LCD\LED 模块

2.17.1 LL_LCDLED_SetMode

2.17.1.1 功能介绍

设置 LCD\LED 模式。

2.17.1.2 接口定义

函数接口	void LL_LCDLED_SetMode (uint32_t ModeSel)
输入	uint32_t ModeSel: { LL_LCDLED_MODE_LED_ARRAY, LL_LCDLED_MODE_LED_DOT, LL_LCDLED_MODE_LCD }
输出	无
返回值	无
资源使用	
说明	

2.17.2 LL_LCDLED_GetMode

2.17.2.1 功能介绍

获取 LCD\LED 模式。

2.17.2.2 接口定义

函数接口	uint32_t LL_LCDLED_GetMode (void)
输入	无
输出	无
返回值	{ LL_LCDLED_MODE_LED_ARRAY, LL_LCDLED_MODE_LED_DOT, LL_LCDLED_MODE_LCD }
资源使用	
说明	

2.17.3 L LL_LCDLED_SetDuty

2.17.3.1 功能介绍

设置 LCD\LED 占空比。

2.17.3.2 接口定义

函数接口	void LL_LCDLED_SetDuty (uint32_t DutyVal)
输入	uint32_t DutyVal: { In LCD MODE: LL_LCDLED_LCD_DUTY0, LL_LCDLED_LCD_DUTY1, LL_LCDLED_LCD_DUTY2, LL_LCDLED_LCD_DUTY3, LL_LCDLED_LCD_DUTY4, LL_LCDLED_LCD_DUTY5, LL_LCDLED_LCD_DUTY6, LL_LCDLED_LCD_DUTY7, In LED DOT MODE: LL_LCDLED_LED_DOT_DUTY0, LL_LCDLED_LED_DOT_DUTY1, LL_LCDLED_LED_DOT_DUTY2, LL_LCDLED_LED_DOT_DUTY3, LL_LCDLED_LED_DOT_DUTY4, LL_LCDLED_LED_DOT_DUTY5, In LED ARRAY MODE: LL_LCDLED_LED_ARRAY_DUTY0, LL_LCDLED_LED_ARRAY_DUTY1, LL_LCDLED_LED_ARRAY_DUTY2, LL_LCDLED_LED_ARRAY_DUTY3, LL_LCDLED_LED_ARRAY_DUTY4, LL_LCDLED_LED_ARRAY_DUTY5, LL_LCDLED_LED_ARRAY_DUTY6, LL_LCDLED_LED_ARRAY_DUTY7 }
输出	无
返回值	无
资源使用	
说明	

2.17.4 L LL_LCDLED_GetDuty

2.17.4.1 功能介绍

获取 LCD\LED 占空比。

2.17.4.2 接口定义

函数接口	uint32_t LL_LCDLED_GetDuty (void)
输入	无
输出	无
返回值	{ In LCD MODE: LL_LCDLED_LCD_DUTY0, LL_LCDLED_LCD_DUTY1, LL_LCDLED_LCD_DUTY2, LL_LCDLED_LCD_DUTY3, LL_LCDLED_LCD_DUTY4, LL_LCDLED_LCD_DUTY5, LL_LCDLED_LCD_DUTY6, LL_LCDLED_LCD_DUTY7, In LED DOT MODE: LL_LCDLED_LED_DOT_DUTY0, LL_LCDLED_LED_DOT_DUTY1,

	LL_LCDLED_LED_DOT_DUTY2, LL_LCDLED_LED_DOT_DUTY3, LL_LCDLED_LED_DOT_DUTY4, LL_LCDLED_LED_DOT_DUTY5, In LED ARRAY MODE: LL_LCDLED_LED_ARRAY_DUTY0, LL_LCDLED_LED_ARRAY_DUTY1, LL_LCDLED_LED_ARRAY_DUTY2, LL_LCDLED_LED_ARRAY_DUTY3, LL_LCDLED_LED_ARRAY_DUTY4, LL_LCDLED_LED_ARRAY_DUTY5, LL_LCDLED_LED_ARRAY_DUTY6, LL_LCDLED_LED_ARRAY_DUTY7 }
资源使用	
说明	

2.17.5 LL_LCDLED_SetBrightLevel

2.17.5.1 功能介绍

设置 LCDLED 亮度等级。

2.17.5.2 接口定义

函数接口	void LL_LCDLED_SetBrightLevel (uint32_t BrightLevel)
输入	uint32_t BrightLevel: { LL_LCDLED_BRIGHTLEVEL_0, LL_LCDLED_BRIGHTLEVEL_1, LL_LCDLED_BRIGHTLEVEL_2, LL_LCDLED_BRIGHTLEVEL_3, LL_LCDLED_BRIGHTLEVEL_4, LL_LCDLED_BRIGHTLEVEL_5, LL_LCDLED_BRIGHTLEVEL_6, LL_LCDLED_BRIGHTLEVEL_7 }
输出	无
返回值	无
资源使用	
说明	

2.17.6 LL_LCDLED_GetBrightLevel

2.17.6.1 功能介绍

获取 LCDLED 亮度等级。

2.17.6.2 接口定义

函数接口	uint32_t LL_LCDLED_GetBrightLevel (void)
输入	无
输出	无
返回值	{ LL_LCDLED_BRIGHTLEVEL_0, LL_LCDLED_BRIGHTLEVEL_1, LL_LCDLED_BRIGHTLEVEL_2, LL_LCDLED_BRIGHTLEVEL_3, LL_LCDLED_BRIGHTLEVEL_4, LL_LCDLED_BRIGHTLEVEL_5, LL_LCDLED_BRIGHTLEVEL_6, LL_LCDLED_BRIGHTLEVEL_7 }
资源使用	
说明	

2.17.7 LL_LCDLED_LED_SetBrightMode

2.17.7.1 功能介绍

设置 LED 亮度模式。

2.17.7.2 接口定义

函数接口	void LL_LCDLED_LED_SetBrightMode (uint32_t BrightMode)
输入	uint32_t BrightMode: { LL_LCDLED_LED_BRIGHTMODE_EVEN, LL_LCDLED_LED_BRIGHTMODE_ENHANCE }
输出	无
返回值	无
资源使用	
说明	

2.17.8 LL_LCDLED_LED_GetBrightMode

2.17.8.1 功能介绍

获取 LED 亮度模式。

2.17.8.2 接口定义

函数接口	uint32_t LL_LCDLED_LED_GetBrightMode (void)
输入	无
输出	无
返回值	{ LL_LCDLED_LED_BRIGHTMODE_EVEN, LL_LCDLED_LED_BRIGHTMODE_ENHANCE }
资源使用	
说明	

2.17.9 LL_LCDLED_EnableClockHold

2.17.9.1 功能介绍

使能 LED 时钟保持。

2.17.9.2 接口定义

函数接口	void LL_LCDLED_EnableClockHold (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.17.10 LL_LCDLED_DisableClockHold

2.17.10.1 功能介绍

禁止使能 LED 时钟保持。

2.17.10.2 接口定义

函数接口	void LL_LCDLED_DisableClockHold (void)
------	--

输入	无
输出	无
返回值	无
资源使用	
说明	

2.17.11 LL_LCDLED_IsEnabledClockHold

2.17.11.1 功能介绍

检查是否使能 LED 时钟保持。

2.17.11.2 接口定义

函数接口	uint32_t LL_LCDLED_IsEnabledClockHold (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.17.12 LL_LCDLED_SetClockDivision

2.17.12.1 功能介绍

设置 LCDLED 时钟分频。

2.17.12.2 接口定义

函数接口	void LL_LCDLED_SetClockDivision (uint32_t ClockDiv)
输入	uint32_t ClockDiv: { LL_LCDLED_CLOCK_DIV_64, LL_LCDLED_CLOCK_DIV_32, LL_LCDLED_CLOCK_DIV_16, LL_LCDLED_CLOCK_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.17.13 LL_LCDLED_GetClockDivision

2.17.13.1 功能介绍

获取 LCDLED 时钟分频。

2.17.13.2 接口定义

函数接口	uint32_t LL_LCDLED_GetClockDivision (void)
输入	无
输出	无
返回值	{ LL_LCDLED_CLOCK_DIV_64, LL_LCDLED_CLOCK_DIV_32, LL_LCDLED_CLOCK_DIV_16, LL_LCDLED_CLOCK_DIV_8 }
资源使用	
说明	

2.17.14 LL_LCDLED_LCD_SetClockSource

2.17.14.1 功能介绍

设置 LCD\LED 时钟源。

2.17.14.2 接口定义

函数接口	void LL_LCDLED_LCD_SetClockSource (uint32_t ClockSource)
输入	uint32_t ClockSource: { LL_LCDLED_CLOCK_DIV_64, LL_LCDLED_CLOCK_DIV_32, LL_LCDLED_CLOCK_DIV_16, LL_LCDLED_CLOCK_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.17.15 LL_LCDLED_LCD_GetClockSource

2.17.15.1 功能介绍

获取 LCD\LED 时钟源。

2.17.15.2 接口定义

函数接口	uint32_t LL_LCDLED_LCD_GetClockSource (void)
输入	无
输出	无
返回值	{ LL_LCDLED_CLOCK_DIV_64, LL_LCDLED_CLOCK_DIV_32, LL_LCDLED_CLOCK_DIV_16, LL_LCDLED_CLOCK_DIV_8 }
资源使用	
说明	

2.17.16 LL_LCDLED_Enable

2.17.16.1 功能介绍

使能 LCD\LED。

2.17.16.2 接口定义

函数接口	void LL_LCDLED_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.17.17 LL_LCDLED_Disable

2.17.17.1 功能介绍

禁止使能 LCD\LED。

2.17.17.2 接口定义

函数接口	void LL_LCDLED_Disable (void)
------	---------------------------------

输入	无
输出	无
返回值	无
资源使用	
说明	

2.17.18 LL_LCDLED_IsEnabled

2.17.18.1 功能介绍

检查是否使能 LCD\LED。

2.17.18.2 接口定义

函数接口	uint32_t LL_LCDLED_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.17.19 LL_LCDLED_SetRAMData

2.17.19.1 功能介绍

设置 LCD\LED RAM 值。

2.17.19.2 接口定义

函数接口	void LL_LCDLED_SetRAMData (uint32_t RAMx, uint32_t Data)
输入	uint32_t RAMx: { LL_LCDLED_RAM_0, LL_LCDLED_RAM_1, LL_LCDLED_RAM_2, LL_LCDLED_RAM_3, LL_LCDLED_RAM_4, LL_LCDLED_RAM_5, LL_LCDLED_RAM_6, LL_LCDLED_RAM_7 } uint32_t Data: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.17.20 LL_LCDLED_GetRAMData

2.17.20.1 功能介绍

获取 LCD\LED RAM 值。

2.17.20.2 接口定义

函数接口	uint32_t LL_LCDLED_GetRAMData (uint32_t RAMx)
输入	uint32_t RAMx: { LL_LCDLED_RAM_0, LL_LCDLED_RAM_1, LL_LCDLED_RAM_2, LL_LCDLED_RAM_3, LL_LCDLED_RAM_4, LL_LCDLED_RAM_5, LL_LCDLED_RAM_6, LL_LCDLED_RAM_7 }

输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.18 LPTIM 模块

属性	类型	字段名	含义
LPTIM_TypeDef			
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断清零寄存器
读写	uint32_t	IER	中断使能寄存器
读写	uint32_t	CFGR	配置寄存器
读写	uint32_t	CR	控制寄存器
读写	uint32_t	CMP	比较寄存器
读写	uint32_t	ARR	自动重载寄存器
只读	uint32_t	CNT	计数器寄存器
读写	uint32_t	CFGR2	配置寄存器 2
LL_LPTIM_InitTypeDef			
读写	uint32_t	ClockSource	时钟源
读写	uint32_t	Prescaler	分频率
读写	uint32_t	Waveform	控制输出波形
读写	uint32_t	Polarity	波形极性

2.18.1 LL_LPTIM_DeInit

2.18.1.1 功能介绍

清除 LPTIM 初始化参数。

2.18.1.2 接口定义

函数接口	ErrorStatus LL_LPTIM_DeInit (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.18.2 LL_LPTIM_StructInit

2.18.2.1 功能介绍

LPTIM 结构体初始化。

2.18.2.2 接口定义

函数接口	void LL_LPTIM_StructInit (LL_LPTIM_InitTypeDef * LPTIM_InitStruct)
输入	LL_LPTIM_InitTypeDef * LPTIM_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.18.3 LL_LPTIM_Init

2.18.3.1 功能介绍

LPTIM 初始化。

2.18.3.2 接口定义

函数接口	ErrorStatus LL_LPTIM_Init (LPTIM_TypeDef * LPTIMx, LL_LPTIM_InitTypeDef * LPTIM_InitStruct)
输入	LPTIM_TypeDef * LPTIMx LL_LPTIM_InitTypeDef * LPTIM_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.18.4 LL_LPTIM_Enable

2.18.4.1 功能介绍

使能 LPTIM。

2.18.4.2 接口定义

函数接口	void LL_LPTIM_Enable (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.5 LL_LPTIM_Disable

2.18.5.1 功能介绍

禁止使能 LPTIM。

2.18.5.2 接口定义

函数接口	void LL_LPTIM_Disable (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无

资源使用	
说明	

2.18.6 LL_LPTIM_IsEnabled

2.18.6.1 功能介绍

检查是否使能 LPTIM。

2.18.6.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabled (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.7 LL_LPTIM_StartCounter

2.18.7.1 功能介绍

启动 LPTIM 计数器模式。

2.18.7.2 接口定义

函数接口	void LL_LPTIM_StartCounter (LPTIM_TypeDef * LPTIMx, uint32_t OperatingMode)
输入	LPTIM_TypeDef * LPTIMx uint32_t OperatingMode: { LL_LPTIM_OPERATING_MODE_CONTINUOUS, LL_LPTIM_OPERATING_MODE_ONESHOT }
输出	无
返回值	无
资源使用	
说明	

2.18.8 LL_LPTIM_EnableResetAfterRead

2.18.8.1 功能介绍

读使能后复位。

2.18.8.2 接口定义

函数接口	void LL_LPTIM_EnableResetAfterRead (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.9 LL_LPTIM_DisableResetAfterRead

2.18.9.1 功能介绍

禁止读使能后复位。

2.18.9.2 接口定义

函数接口	void LL_LPTIM_DisableResetAfterRead (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.10 LL_LPTIM_IsEnabledResetAfterRead

2.18.10.1 功能介绍

检查是否读使能后复位。

2.18.10.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledResetAfterRead (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.11 LL_LPTIM_ResetCounter

2.18.11.1 功能介绍

重置 LPTIM 计数器。

2.18.11.2 接口定义

函数接口	void LL_LPTIM_ResetCounter (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.12 LL_LPTIM_SetUpdateMode

2.18.12.1 功能介绍

设置 LPTIM 寄存器更新模式。

2.18.12.2 接口定义

函数接口	void LL_LPTIM_SetUpdateMode (LPTIM_TypeDef * LPTIMx, uint32_t UpdateMode)
输入	LPTIM_TypeDef * LPTIMx

	uint32_t UpdateMode: { LL_LPTIM_UPDATE_MODE_IMMEDIATE, LL_LPTIM_UPDATE_MODE_ENDOFPERIOD }
输出	无
返回值	无
资源使用	
说明	

2.18.13 LL_LPTIM_GetUpdateMode

2.18.13.1 功能介绍

获取 LPTIM 寄存器更新模式。

2.18.13.2 接口定义

函数接口	uint32_t LL_LPTIM_GetUpdateMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_UPDATE_MODE_IMMEDIATE, LL_LPTIM_UPDATE_MODE_ENDOFPERIOD }
资源使用	
说明	

2.18.14 LL_LPTIM_SetAutoReload

2.18.14.1 功能介绍

设置 LPTIM 自动重载值。

2.18.14.2 接口定义

函数接口	void LL_LPTIM_SetAutoReload (LPTIM_TypeDef * LPTIMx, uint32_t AutoReload)
输入	LPTIM_TypeDef * LPTIMx uint32_t AutoReload: [0x0000, 0xFFFF]
输出	无
返回值	无
资源使用	
说明	

2.18.15 LL_LPTIM_GetAutoReload

2.18.15.1 功能介绍

获取 LPTIM 自动重载值。

2.18.15.2 接口定义

函数接口	uint32_t LL_LPTIM_GetAutoReload (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	[0x0000, 0xFFFF]

资源使用	
说明	

2.18.16 LL_LPTIM_SetCompare

2.18.16.1 功能介绍

设置 LPTIM 比较值。

2.18.16.2 接口定义

函数接口	void LL_LPTIM_SetCompare (LPTIM_TypeDef * LPTIMx, uint32_t CompareValue)
输入	LPTIM_TypeDef * LPTIMx uint32_t CompareValue: [0x0000, 0xFFFF]
输出	无
返回值	无
资源使用	
说明	

2.18.17 LL_LPTIM_GetCompare

2.18.17.1 功能介绍

获取 LPTIM 比较值。

2.18.17.2 接口定义

函数接口	uint32_t LL_LPTIM_GetCompare (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	[0x0000, 0xFFFF]
资源使用	
说明	

2.18.18 LL_LPTIM_GetCounter

2.18.18.1 功能介绍

获取 LPTIM 计数值。

2.18.18.2 接口定义

函数接口	uint32_t LL_LPTIM_GetCounter (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	[0x0000, 0xFFFF]
资源使用	
说明	

2.18.19 LL_LPTIM_SetCounterMode

2.18.19.1 功能介绍

设置 LPTIM 计数模式。

2.18.19.2 接口定义

函数接口	void LL_LPTIM_SetCounterMode (LPTIM_TypeDef * LPTIMx, uint32_t CounterMode)
输入	LPTIM_TypeDef * LPTIMx uint32_t CounterMode: { LL_LPTIM_COUNTER_MODE_INTERNAL, LL_LPTIM_COUNTER_MODE_EXTERNAL }
输出	无
返回值	无
资源使用	
说明	

2.18.20 LL_LPTIM_GetCounterMode

2.18.20.1 功能介绍

获取 LPTIM 计数模式。

2.18.20.2 接口定义

函数接口	uint32_t LL_LPTIM_GetCounterMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_COUNTER_MODE_INTERNAL, LL_LPTIM_COUNTER_MODE_EXTERNAL }
资源使用	
说明	

2.18.21 LL_LPTIM_ConfigOutput

2.18.21.1 功能介绍

配置 LPTIM 输出模式。

2.18.21.2 接口定义

函数接口	void LL_LPTIM_ConfigOutput (LPTIM_TypeDef * LPTIMx, uint32_t Waveform, uint32_t Polarity)
输入	LPTIM_TypeDef * LPTIMx uint32_t Waveform: {LL_LPTIM_OUTPUT_WAVEFORM_PWM, LL_LPTIM_OUTPUT_WAVEFORM_SETONCE } uint32_t Polarity: {LL_LPTIM_OUTPUT_POLARITY_REGULAR, LL_LPTIM_OUTPUT_POLARITY_INVERSE }
输出	无
返回值	无
资源使用	
说明	

2.18.22 LL_LPTIM_SetWaveform

2.18.22.1 功能介绍

设置 LPTIM 波形模式。

2.18.22.2 接口定义

函数接口	void LL_LPTIM_SetWaveform (LPTIM_TypeDef * LPTIMx, uint32_t Waveform)
输入	LPTIM_TypeDef * LPTIMx uint32_t Waveform: {LL_LPTIM_OUTPUT_WAVEFORM_PWM, LL_LPTIM_OUTPUT_WAVEFORM_SETONCE }
输出	无
返回值	无
资源使用	
说明	

2.18.23 LL_LPTIM_GetWaveform

2.18.23.1 功能介绍

获取 LPTIM 波形模式。

2.18.23.2 接口定义

函数接口	uint32_t LL_LPTIM_GetWaveform (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{LL_LPTIM_OUTPUT_WAVEFORM_PWM, LL_LPTIM_OUTPUT_WAVEFORM_SETONCE }
资源使用	
说明	

2.18.24 LL_LPTIM_SetPolarity

2.18.24.1 功能介绍

设置 LPTIM 极性模式。

2.18.24.2 接口定义

函数接口	void LL_LPTIM_SetPolarity (LPTIM_TypeDef * LPTIMx, uint32_t Polarity)
输入	LPTIM_TypeDef * LPTIMx uint32_t Polarity: {LL_LPTIM_OUTPUT_POLARITY_REGULAR, LL_LPTIM_OUTPUT_POLARITY_INVERSE }
输出	无
返回值	无
资源使用	
说明	

2.18.25 LL_LPTIM_GetPolarity

2.18.25.1 功能介绍

获取 LPTIM 极性模式。

2.18.25.2 接口定义

函数接口	uint32_t LL_LPTIM_GetPolarity (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{LL_LPTIM_OUTPUT_POLARITY_REGULAR, LL_LPTIM_OUTPUT_POLARITY_INVERSE }
资源使用	
说明	

2.18.26 LL_LPTIM_SetPrescaler

2.18.26.1 功能介绍

设置 LPTIM 分频率。

2.18.26.2 接口定义

函数接口	void LL_LPTIM_SetPrescaler (LPTIM_TypeDef * LPTIMx, uint32_t Prescaler)
输入	LPTIM_TypeDef * LPTIMx uint32_t Prescaler: { LL_LPTIM_PRESCALER_DIV1, LL_LPTIM_PRESCALER_DIV2, LL_LPTIM_PRESCALER_DIV4, LL_LPTIM_PRESCALER_DIV8, LL_LPTIM_PRESCALER_DIV16, LL_LPTIM_PRESCALER_DIV32, LL_LPTIM_PRESCALER_DIV64, LL_LPTIM_PRESCALER_DIV128 }
输出	无
返回值	无
资源使用	
说明	

2.18.27 LL_LPTIM_GetPrescaler

2.18.27.1 功能介绍

获取 LPTIM 分频率。

2.18.27.2 接口定义

函数接口	uint32_t LL_LPTIM_GetPrescaler (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_PRESCALER_DIV1, LL_LPTIM_PRESCALER_DIV2, LL_LPTIM_PRESCALER_DIV4, LL_LPTIM_PRESCALER_DIV8, LL_LPTIM_PRESCALER_DIV16, LL_LPTIM_PRESCALER_DIV32, LL_LPTIM_PRESCALER_DIV64, LL_LPTIM_PRESCALER_DIV128 }
资源使用	

说明	
----	--

2.18.28 LL_LPTIM_SetInput1Polarity

2.18.28.1 功能介绍

设置 LPTIM_IN1 输入极性。

2.18.28.2 接口定义

函数接口	void LL_LPTIM_SetInput1Polarity (LPTIM_TypeDef * LPTIMx, uint32_t Polarity)
输入	LPTIM_TypeDef * LPTIMx uint32_t Polarity: { LL_LPTIM_INPUT1_POLARITY_REGULAR, LL_LPTIM_INPUT1_POLARITY_INVERSE }
输出	无
返回值	无
资源使用	
说明	

2.18.29 LL_LPTIM_GetInput1Polarity

2.18.29.1 功能介绍

获取 LPTIM_IN1 输入极性。

2.18.29.2 接口定义

函数接口	uint32_t LL_LPTIM_GetInput1Polarity (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_INPUT1_POLARITY_REGULAR, LL_LPTIM_INPUT1_POLARITY_INVERSE }
资源使用	
说明	

2.18.30 LL_LPTIM_SetInput2Polarity

2.18.30.1 功能介绍

设置 LPTIM_IN2 输入极性。

2.18.30.2 接口定义

函数接口	void LL_LPTIM_SetInput2Polarity (LPTIM_TypeDef * LPTIMx, uint32_t Polarity)
输入	LPTIM_TypeDef * LPTIMx uint32_t Polarity: { LL_LPTIM_INPUT2_POLARITY_REGULAR, LL_LPTIM_INPUT2_POLARITY_INVERSE }
输出	无
返回值	无
资源使用	

说明	
----	--

2.18.31 LL_LPTIM_GetInput2Polarity

2.18.31.1 功能介绍

获取 LPTIM_IN2 输入极性。

2.18.31.2 接口定义

函数接口	uint32_t LL_LPTIM_GetInput2Polarity (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_INPUT2_POLARITY_REGULAR, LL_LPTIM_INPUT2_POLARITY_INVERSE }
资源使用	
说明	

2.18.32 LL_LPTIM_SetInput1LSI

2.18.32.1 功能介绍

设置 LPTIM input 1 选择信号。

2.18.32.2 接口定义

函数接口	void LL_LPTIM_SetInput1LSI (LPTIM_TypeDef * LPTIMx, uint32_t LSI)
输入	LPTIM_TypeDef * LPTIMx uint32_t LSI: { LL_LPTIM_INPUT1_LSI_GPIO, LL_LPTIM_INPUT1_LSI_COMP1, LL_LPTIM_INPUT1_LSI_COMP3, LL_LPTIM_INPUT1_LSI_COMP1_COMP2 }
输出	无
返回值	无
资源使用	
说明	

2.18.33 LL_LPTIM_GetInput1LSI

2.18.33.1 功能介绍

获取 LPTIM input 1 选择信号。

2.18.33.2 接口定义

函数接口	uint32_t LL_LPTIM_GetInput1LSI (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_INPUT1_LSI_GPIO, LL_LPTIM_INPUT1_LSI_COMP1, LL_LPTIM_INPUT1_LSI_COMP3, LL_LPTIM_INPUT1_LSI_COMP1_COMP2 }
资源使用	
说明	

2.18.34 LL_LPTIM_SetInput2LSI

2.18.34.1 功能介绍

设置 LPTIM input 2 选择信号。

2.18.34.2 接口定义

函数接口	void LL_LPTIM_SetInput2LSI (LPTIM_TypeDef * LPTIMx, uint32_t LSI)
输入	LPTIM_TypeDef * LPTIMx uint32_t LSI: { LL_LPTIM_INPUT2_LSI_GPIO, LL_LPTIM_INPUT2_LSI_COMP2, LL_LPTIM_INPUT2_LSI_COMP4, LL_LPTIM_INPUT2_LSI_COMP3_COMP4 }
输出	无
返回值	无
资源使用	
说明	

2.18.35 LL_LPTIM_GetInput2LSI

2.18.35.1 功能介绍

获取 LPTIM input 2 选择信号。

2.18.35.2 接口定义

函数接口	uint32_t LL_LPTIM_GetInput2LSI (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_INPUT2_LSI_GPIO, LL_LPTIM_INPUT2_LSI_COMP2, LL_LPTIM_INPUT2_LSI_COMP4, LL_LPTIM_INPUT2_LSI_COMP3_COMP4 }
资源使用	
说明	

2.18.36 LL_LPTIM_EnableTimeout

2.18.36.1 功能介绍

使能超时功能。

2.18.36.2 接口定义

函数接口	void LL_LPTIM_EnableTimeout (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.37 LL_LPTIM_DisableTimeout

2.18.37.1 功能介绍

禁止使能超时功能。

2.18.37.2 接口定义

函数接口	void LL_LPTIM_DisableTimeout (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.38 LL_LPTIM_IsEnabledTimeout

2.18.38.1 功能介绍

检查是否使能超时功能。

2.18.38.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledTimeout (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.39 LL_LPTIM_TrigSw

2.18.39.1 功能介绍

设置 LPTIM 软件触发模式。

2.18.39.2 接口定义

函数接口	void LL_LPTIM_TrigSw (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.40 LL_LPTIM_ConfigTrigger

2.18.40.1 功能介绍

配置外部触发器，用作 LPTIM 的触发器事件。

2.18.40.2 接口定义

函数接口	void LL_LPTIM_ConfigTrigger (LPTIM_TypeDef * LPTIMx, uint32_t Source, uint32_t Filter, uint32_t Polarity)
输入	LPTIM_TypeDef * LPTIMx uint32_t Source:

	{ LL_LPTIM_TRIGSOURCE_0, LL_LPTIM_TRIGSOURCE_1, LL_LPTIM_TRIGSOURCE_4, LL_LPTIM_TRIGSOURCE_5, LL_LPTIM_TRIGSOURCE_6, LL_LPTIM_TRIGSOURCE_7 } uint32_t Filter: { LL_LPTIM_TRIG_FILTER_NONE, LL_LPTIM_TRIG_FILTER_2, LL_LPTIM_TRIG_FILTER_4, LL_LPTIM_TRIG_FILTER_8 } uint32_t Polarity: { LL_LPTIM_TRIG_POLARITY_RISING, LL_LPTIM_TRIG_POLARITY_FALLING, LL_LPTIM_TRIG_POLARITY_RISING_FALLING }
输出	无
返回值	无
资源使用	
说明	

2.18.41 LL_LPTIM_SetTriggerFilter

2.18.41.1 功能介绍

设置 LPTIM 触发源滤波。

2.18.41.2 接口定义

函数接口	void LL_LPTIM_SetTriggerFilter (LPTIM_TypeDef * LPTIMx, uint32_t Filter)
输入	LPTIM_TypeDef * LPTIMx uint32_t Filter: { LL_LPTIM_TRIG_FILTER_NONE, LL_LPTIM_TRIG_FILTER_2, LL_LPTIM_TRIG_FILTER_4, LL_LPTIM_TRIG_FILTER_8 }
输出	无
返回值	无
资源使用	
说明	

2.18.42 LL_LPTIM_GetTriggerFilter

2.18.42.1 功能介绍

获取 LPTIM 触发源滤波。

2.18.42.2 接口定义

函数接口	uint32_t LL_LPTIM_GetTriggerFilter (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_TRIG_FILTER_NONE, LL_LPTIM_TRIG_FILTER_2, LL_LPTIM_TRIG_FILTER_4, LL_LPTIM_TRIG_FILTER_8 }
资源使用	
说明	

2.18.43 LL_LPTIM_GetTriggerSource

2.18.43.1 功能介绍

获取 LPTIM 触发源。

2.18.43.2 接口定义

函数接口	uint32_t LL_LPTIM_GetTriggerSource (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_TRIGSOURCE_0, LL_LPTIM_TRIGSOURCE_1, LL_LPTIM_TRIGSOURCE_4, LL_LPTIM_TRIGSOURCE_5, LL_LPTIM_TRIGSOURCE_6, LL_LPTIM_TRIGSOURCE_7 }
资源使用	
说明	

2.18.44 LL_LPTIM_GetTriggerPolarity

2.18.44.1 功能介绍

获取 LPTIM 触发源极性。

2.18.44.2 接口定义

函数接口	uint32_t LL_LPTIM_GetTriggerPolarity (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_TRIG_POLARITY_RISING, LL_LPTIM_TRIG_POLARITY_FALLING, LL_LPTIM_TRIG_POLARITY_RISING_FALLING }
资源使用	
说明	

2.18.45 LL_LPTIM_SetClockSource

2.18.45.1 功能介绍

设置 LPTIM 时钟源。

2.18.45.2 接口定义

函数接口	void LL_LPTIM_SetClockSource (LPTIM_TypeDef * LPTIMx, uint32_t ClockSource)
输入	LPTIM_TypeDef * LPTIMx uint32_t ClockSource: { LL_LPTIM_CLK_SOURCE_INTERNAL, LL_LPTIM_CLK_SOURCE_EXTERNAL }
输出	无
返回值	无
资源使用	
说明	

2.18.46 LL_LPTIM_GetClockSource

2.18.46.1 功能介绍

获取 LPTIM 时钟源。

2.18.46.2 接口定义

函数接口	uint32_t LL_LPTIM_GetClockSource (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_CLK_SOURCE_INTERNAL, LL_LPTIM_CLK_SOURCE_EXTERNAL }
资源使用	
说明	

2.18.47 LL_LPTIM_ConfigClock

2.18.47.1 功能介绍

配置 LPTIM 时钟。

2.18.47.2 接口定义

函数接口	void LL_LPTIM_ConfigClock (LPTIM_TypeDef * LPTIMx, uint32_t ClockFilter, uint32_t ClockPolarity, uint32_t InputFilterDiv)
输入	LPTIM_TypeDef * LPTIMx uint32_t ClockFilter: { LL_LPTIM_CLK_FILTER_NONE, LL_LPTIM_CLK_FILTER_2, LL_LPTIM_CLK_FILTER_4, LL_LPTIM_CLK_FILTER_8 } uint32_t ClockPolarity: { LL_LPTIM_CLK_POLARITY_RISING, LL_LPTIM_CLK_POLARITY_FALLING, LL_LPTIM_CLK_POLARITY_RISING_FALLING } uint32_t InputFilterDiv: { LL_LPTIM_FILTER_CLK_DIV1, LL_LPTIM_FILTER_CLK_DIV2, LL_LPTIM_FILTER_CLK_DIV4, LL_LPTIM_FILTER_CLK_DIV8, LL_LPTIM_FILTER_CLK_DIV16, LL_LPTIM_FILTER_CLK_DIV32, LL_LPTIM_FILTER_CLK_DIV64, LL_LPTIM_FILTER_CLK_DIV128 }
输出	无
返回值	无
资源使用	
说明	

2.18.48 LL_LPTIM_SetClockFilter

2.18.48.1 功能介绍

设置实际时钟数字滤波器。

2.18.48.2 接口定义

函数接口	void LL_LPTIM_SetClockFilter (LPTIM_TypeDef * LPTIMx, uint32_t ClockFilter)
------	---

输入	LPTIM_TypeDef * LPTIMx uint32_t ClockFilter: { LL_LPTIM_CLK_FILTER_NONE, LL_LPTIM_CLK_FILTER_2, LL_LPTIM_CLK_FILTER_4, LL_LPTIM_CLK_FILTER_8 }
输出	无
返回值	无
资源使用	
说明	

2.18.49 LL_LPTIM_GetClockFilter

2.18.49.1 功能介绍

获取实际时钟数字滤波器。

2.18.49.2 接口定义

函数接口	uint32_t LL_LPTIM_GetClockFilter (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_CLK_FILTER_NONE, LL_LPTIM_CLK_FILTER_2, LL_LPTIM_CLK_FILTER_4, LL_LPTIM_CLK_FILTER_8 }
资源使用	
说明	

2.18.50 LL_LPTIM_GetClockPolarity

2.18.50.1 功能介绍

获取实际的时钟极性。

2.18.50.2 接口定义

函数接口	uint32_t LL_LPTIM_GetClockPolarity (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_CLK_POLARITY_RISING, LL_LPTIM_CLK_POLARITY_FALLING, LL_LPTIM_CLK_POLARITY_RISING_FALLING }
资源使用	
说明	

2.18.51 LL_LPTIM_GetFilterClockDiv

2.18.51.1 功能介绍

获取实际的数字滤波器时钟分频。。

2.18.51.2 接口定义

函数接口	uint32_t LL_LPTIM_GetFilterClockDiv (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_FILTER_CLK_DIV1, LL_LPTIM_FILTER_CLK_DIV2,

	LL_LPTIM_FILTER_CLK_DIV4, LL_LPTIM_FILTER_CLK_DIV8, LL_LPTIM_FILTER_CLK_DIV16, LL_LPTIM_FILTER_CLK_DIV32, LL_LPTIM_FILTER_CLK_DIV64, LL_LPTIM_FILTER_CLK_DIV128 }
资源使用	
说明	

2.18.52 LL_LPTIM_SetEncoderMode

2.18.52.1 功能介绍

配置正交编码器模式。

2.18.52.2 接口定义

函数接口	void LL_LPTIM_SetEncoderMode (LPTIM_TypeDef * LPTIMx, uint32_t EncoderMode)
输入	LPTIM_TypeDef * LPTIMx uint32_t EncoderMode: { LL_LPTIM_ENCODER_MODE_RISING, LL_LPTIM_ENCODER_MODE_FALLING, LL_LPTIM_ENCODER_MODE_RISING_FALLING }
输出	无
返回值	无
资源使用	
说明	

2.18.53 LL_LPTIM_GetEncoderMode

2.18.53.1 功能介绍

获取正交编码器模式。

2.18.53.2 接口定义

函数接口	uint32_t LL_LPTIM_GetEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_ENCODER_MODE_RISING, LL_LPTIM_ENCODER_MODE_FALLING, LL_LPTIM_ENCODER_MODE_RISING_FALLING }
资源使用	
说明	

2.18.54 LL_LPTIM_EnableEncoderMode

2.18.54.1 功能介绍

使能正交编码器模式。

2.18.54.2 接口定义

函数接口	void LL_LPTIM_EnableEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无

返回值	无
资源使用	
说明	

2.18.55 LL_LPTIM_DisableEncoderMode

2.18.55.1 功能介绍

禁止使能正交编码器模式。

2.18.55.2 接口定义

函数接口	void LL_LPTIM_DisableEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.56 LL_LPTIM_IsEnabledEncoderMode

2.18.56.1 功能介绍

检查是否使能正交编码器模式。

2.18.56.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.57 LL_LPTIM_SetDualEncoderMode

2.18.57.1 功能介绍

设置非交编码器模式方向。

2.18.57.2 接口定义

函数接口	void LL_LPTIM_SetDualEncoderMode (LPTIM_TypeDef * LPTIMx, uint32_t DualEncoderMode)
输入	LPTIM_TypeDef * LPTIMx uint32_t DualEncoderMode: { LL_LPTIM_DUALENCODER_MODE_RISING, LL_LPTIM_DUALENCODER_MODE_FALLING }
输出	无
返回值	无
资源使用	
说明	

2.18.58 LL_LPTIM_GetDualEncoderMode

2.18.58.1 功能介绍

获取非交编码器模式方向。

2.18.58.2 接口定义

函数接口	uint32_t LL_LPTIM_GetDualEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{ LL_LPTIM_DUALENCODER_MODE_RISING, LL_LPTIM_DUALENCODER_MODE_FALLING }
资源使用	
说明	

2.18.59 LL_LPTIM_EnableDualEncoderMode

2.18.59.1 功能介绍

使能非交编码器模式。

2.18.59.2 接口定义

函数接口	void LL_LPTIM_EnableDualEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.60 LL_LPTIM_DisableDualEncoderMode

2.18.60.1 功能介绍

禁止使能非交编码器模式。

2.18.60.2 接口定义

函数接口	void LL_LPTIM_DisableDualEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.61 LL_LPTIM_IsEnabledDualEncoderMode

2.18.61.1 功能介绍

检查是否使能非交编码器模式。

2.18.61.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledDualEncoderMode (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.62LL_LPTIM_ClearFLAG_CMPM

2.18.62.1 功能介绍

清除比较匹配标志。

2.18.62.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_CMPM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.63LL_LPTIM_IsActiveFlag_CMPM

2.18.63.1 功能介绍

检查比较匹配标志。

2.18.63.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_CMPM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.64LL_LPTIM_ClearFLAG_ARRM

2.18.64.1 功能介绍

清除自动重载匹配标志。

2.18.64.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_ARRM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.65LL_LPTIM_IsActiveFlag_ARRM

2.18.65.1 功能介绍

检查自动重载匹配标志。

2.18.65.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_ARRM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.66LL_LPTIM_ClearFLAG_EXTTRIG

2.18.66.1 功能介绍

清除外部触发有效边沿标志。

2.18.66.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_EXTTRIG (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.67LL_LPTIM_IsActiveFlag_EXTTRIG

2.18.67.1 功能介绍

检查外部触发有效边沿标志。

2.18.67.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_EXTTRIG (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.68LL_LPTIM_ClearFLAG_CMPOK

2.18.68.1 功能介绍

清除比较寄存器更新成功标志。

2.18.68.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_CMPOK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx

输出	无
返回值	无
资源使用	
说明	

2.18.69LL_LPTIM_IsActiveFlag_CMPOK

2.18.69.1 功能介绍

检查比较寄存器更新成功标志。

2.18.69.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_CMPOK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.70LL_LPTIM_ClearFLAG_ARROK

2.18.70.1 功能介绍

清除自动重载寄存器更新成功标志。

2.18.70.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_ARROK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.71LL_LPTIM_IsActiveFlag_ARROK

2.18.71.1 功能介绍

检查自动重载寄存器更新成功标志。

2.18.71.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_ARROK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.72 LL_LPTIM_ClearFLAG_UP

2.18.72.1 功能介绍

清除计数方向从递减变为递增标志。

2.18.72.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_UP (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.73 LL_LPTIM_IsActiveFlag_UP

2.18.73.1 功能介绍

检查计数方向从递减变为递增标志。

2.18.73.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_UP (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.74 LL_LPTIM_ClearFLAG_DOWN

2.18.74.1 功能介绍

清除方向变为递减标志。

2.18.74.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_DOWN (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.75 LL_LPTIM_IsActiveFlag_DOWN

2.18.75.1 功能介绍

检查方向变为递减标志。

2.18.75.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_DOWN (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx

输出	无
返回值	{0,1}
资源使用	
说明	

2.18.76LL_LPTIM_ClearFLAG_IN1IT

2.18.76.1 功能介绍

清除非交编码模式，通道 1 缺失错误中断标志。

2.18.76.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_IN1IT (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.77LL_LPTIM_IsActiveFlag_IN1IT

2.18.77.1 功能介绍

检查非交编码模式，通道 1 缺失错误中断标志。

2.18.77.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_IN1IT (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.78LL_LPTIM_ClearFLAG_IN2IT

2.18.78.1 功能介绍

清除非交编码模式，通道 2 缺失错误中断标志。

2.18.78.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_IN2IT (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.79LL_LPTIM_IsActiveFlag_IN2IT

2.18.79.1 功能介绍

检查非交编码模式，通道 2 缺失错误中断标志。

2.18.79.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_IN2IT (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.80LL_LPTIM_ClearFLAG_INB2B

2.18.80.1 功能介绍

清除非交编码模式，双通道脉冲连续错误中断标志。

2.18.80.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_INB2B (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.81LL_LPTIM_IsActiveFlag_INB2B

2.18.81.1 功能介绍

检查非交编码模式，双通道脉冲连续错误中断标志。

2.18.81.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_INB2B (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.82LL_LPTIM_ClearFLAG_DUALError

2.18.82.1 功能介绍

清除非交波形错误中断标志。

2.18.82.2 接口定义

函数接口	void LL_LPTIM_ClearFLAG_DUALError (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.83 LL_LPTIM_IsActiveFlag_DUALError

2.18.83.1 功能介绍

检查非交波形错误中断标志。

2.18.83.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag_DUALError (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.84 LL_LPTIM_ClearFLAG

2.18.84.1 功能介绍

清除标志。

2.18.84.2 接口定义

函数接口	void LL_LPTIM_ClearFlag (LPTIM_TypeDef * LPTIMx, uint32_t Flag)
输入	LPTIM_TypeDef * LPTIMx uint32_t Flag: { LL_LPTIM_ISR_CMPM, LL_LPTIM_ISR_CMPOK, LL_LPTIM_ISR_ARRM, LL_LPTIM_ISR_EXTTRIG, LL_LPTIM_ISR_ARROK, LL_LPTIM_ISR_UP, LL_LPTIM_ISR_DOWN, LL_LPTIM_ISR_IN1IDLE, LL_LPTIM_ISR_IN2IDLE, LL_LPTIM_ISR_INB2B, LL_LPTIM_ISR_DUALERROR }
输出	无
返回值	无
资源使用	
说明	

2.18.85 LL_LPTIM_IsActiveFlag

2.18.85.1 功能介绍

检查标志。

2.18.85.2 接口定义

函数接口	uint32_t LL_LPTIM_IsActiveFlag (LPTIM_TypeDef * LPTIMx, uint32_t Flag)
输入	LPTIM_TypeDef * LPTIMx uint32_t Flag: { LL_LPTIM_ISR_CMPM, LL_LPTIM_ISR_CMPOK, LL_LPTIM_ISR_ARRM, LL_LPTIM_ISR_EXTTRIG, LL_LPTIM_ISR_ARROK, LL_LPTIM_ISR_UP, LL_LPTIM_ISR_DOWN, LL_LPTIM_ISR_IN1IDLE,

	LL_LPTIM_ISR_IN2IDLE, LL_LPTIM_ISR_INB2B, LL_LPTIM_ISR_DUALERROR }
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.86 LL_LPTIM_EnableIT_CMPM

2.18.86.1 功能介绍

使能比较匹配中断。

2.18.86.2 接口定义

函数接口	void LL_LPTIM_EnableIT_CMPM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.87 LL_LPTIM_DisableIT_CMPM

2.18.87.1 功能介绍

禁止使能比较匹配中断。

2.18.87.2 接口定义

函数接口	void LL_LPTIM_DisableIT_CMPM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.88 LL_LPTIM_IsEnabledIT_CMPM

2.18.88.1 功能介绍

检查是否使能比较匹配中断。

2.18.88.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_CMPM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.89LL_LPTIM_EnableIT_ARRM

2.18.89.1 功能介绍

使能自动重载匹配中断。

2.18.89.2 接口定义

函数接口	void LL_LPTIM_EnableIT_ARRM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.90LL_LPTIM_DisableIT_ARRM

2.18.90.1 功能介绍

禁止使能自动重载匹配中断。

2.18.90.2 接口定义

函数接口	void LL_LPTIM_DisableIT_ARRM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.91LL_LPTIM_IsEnabledIT_ARRM

2.18.91.1 功能介绍

检查是否使能自动重载匹配中断

2.18.91.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_ARRM (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.92LL_LPTIM_EnableIT_EXTTRIG

2.18.92.1 功能介绍

使能外部触发有效边沿中断。

2.18.92.2 接口定义

函数接口	void LL_LPTIM_EnableIT_EXTTRIG (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无

返回值	无
资源使用	
说明	

2.18.93LL_LPTIM_DisableIT_EXTTRIG

2.18.93.1 功能介绍

禁止使能外部触发有效边沿中断。

2.18.93.2 接口定义

函数接口	void LL_LPTIM_DisableIT_EXTTRIG (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.94LL_LPTIM_IsEnabledIT_EXTTRIG

2.18.94.1 功能介绍

检查是否使能外部触发有效边沿中断。

2.18.94.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_EXTTRIG (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.95LL_LPTIM_EnableIT_CMPMOK

2.18.95.1 功能介绍

使能比较寄存器更新成功中断。

2.18.95.2 接口定义

函数接口	void LL_LPTIM_EnableIT_CMPOK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.96LL_LPTIM_DisableIT_CMPOK

2.18.96.1 功能介绍

禁止使能比较寄存器更新成功中断。

2.18.96.2 接口定义

函数接口	void LL_LPTIM_DisableIT_CMPOK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.97LL_LPTIM_IsEnabledIT_CMPOK

2.18.97.1 功能介绍

检查是否使能比较寄存器更新成功中断。

2.18.97.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_CMPOK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.98LL_LPTIM_EnableIT_ARROK

2.18.98.1 功能介绍

使能自动重载寄存器更新成功中断。

2.18.98.2 接口定义

函数接口	void LL_LPTIM_EnableIT_ARROK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.99LL_LPTIM_DisableIT_ARROK

2.18.99.1 功能介绍

禁止使能自动重载寄存器更新成功中断。

2.18.99.2 接口定义

函数接口	void LL_LPTIM_DisableIT_ARROK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.100 LL_LPTIM_IsEnabledIT_ARROK

2.18.100.1 功能介绍

检查是否使能自动重载寄存器更新成功中断。

2.18.100.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_ARROK (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.101 LL_LPTIM_EnableIT_UP

2.18.101.1 功能介绍

使能方向变为递增中断。

2.18.101.2 接口定义

函数接口	void LL_LPTIM_EnableIT_UP (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.102 LL_LPTIM_DisableIT_UP

2.18.102.1 功能介绍

禁止使能方向变为递增中断。

2.18.102.2 接口定义

函数接口	void LL_LPTIM_DisableIT_UP (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.103 LL_LPTIM_IsEnabledIT_UP

2.18.103.1 功能介绍

检查是否使能方向变为递增中断。

2.18.103.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_UP (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx

输出	无
返回值	{0,1}
资源使用	
说明	

2.18.104 LL_LPTIM_EnableIT_DOWN

2.18.104.1 功能介绍

使能方向变为递减中断。

2.18.104.2 接口定义

函数接口	void LL_LPTIM_EnableIT_DOWN (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.105 LL_LPTIM_DisableIT_DOWN

2.18.105.1 功能介绍

禁止使能方向变为递减中断。

2.18.105.2 接口定义

函数接口	void LL_LPTIM_DisableIT_DOWN (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.106 LL_LPTIM_IsEnabledIT_DOWN

2.18.106.1 功能介绍

检查是否使能方向变为递减中断。

2.18.106.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_DOWN (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.107 LL_LPTIM_EnableIT_IN1

2.18.107.1 功能介绍

使能非交编码模式通道 1 缺失错误中断。

2.18.107.2 接口定义

函数接口	void LL_LPTIM_EnableIT_IN1 (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.108 LL_LPTIM_DisableIT_IN1

2.18.108.1 功能介绍

禁止使能非交编码模式通道 1 缺失错误中断。

2.18.108.2 接口定义

函数接口	void LL_LPTIM_DisableIT_IN1 (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.109 LL_LPTIM_IsEnabledIT_IN1

2.18.109.1 功能介绍

检查是否使能非交编码模式通道 1 缺失错误中断。

2.18.109.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_IN1 (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.110 LL_LPTIM_EnableIT_IN2

2.18.110.1 功能介绍

使能非交编码模式通道 2 缺失错误中断。

2.18.110.2 接口定义

函数接口	void LL_LPTIM_EnableIT_IN2 (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.111 LL_LPTIM_DisableIT_IN2

2.18.111.1 功能介绍

禁止使能非交编码模式通道 2 缺失错误中断。

2.18.111.2 接口定义

函数接口	void LL_LPTIM_DisableIT_IN2 (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.112 LL_LPTIM_IsEnabledIT_IN2

2.18.112.1 功能介绍

检查是否使能非交编码模式通道 2 缺失错误中断。

2.18.112.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_IN2 (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.113 LL_LPTIM_EnableIT_INB2B

2.18.113.1 功能介绍

使能非交编码模式双通道脉冲连续错误中断。

2.18.113.2 接口定义

函数接口	void LL_LPTIM_EnableIT_INB2B (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.114 LL_LPTIM_DisableIT_INB2B

2.18.114.1 功能介绍

禁止使能非交编码模式双通道脉冲连续错误中断。

2.18.114.2 接口定义

函数接口	void LL_LPTIM_DisableIT_INB2B (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无

返回值	无
资源使用	
说明	

2.18.115 LL_LPTIM_IsEnabledIT_INB2B

2.18.115.1 功能介绍

检查是否使能非交编码模式双通道脉冲连续错误中断。

2.18.115.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_INB2B (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.116 LL_LPTIM_EnableIT_DUALError

2.18.116.1 功能介绍

使能非交波形错误中断。

2.18.116.2 接口定义

函数接口	void LL_LPTIM_EnableIT_DUALError (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.117 LL_LPTIM_DisableIT_DUALError

2.18.117.1 功能介绍

禁止使能非交波形错误中断。

2.18.117.2 接口定义

函数接口	void LL_LPTIM_DisableIT_DUALError (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	无
资源使用	
说明	

2.18.118 LL_LPTIM_IsEnabledIT_DUALError

2.18.118.1 功能介绍

检查是否使能非交波形错误中断。

2.18.118.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT_DUALError (LPTIM_TypeDef * LPTIMx)
输入	LPTIM_TypeDef * LPTIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.18.119 LL_LPTIM_EnableIT

2.18.119.1 功能介绍

使能中断。

2.18.119.2 接口定义

函数接口	void LL_LPTIM_EnableIT (LPTIM_TypeDef * LPTIMx, uint32_t Interrupt)
输入	LPTIM_TypeDef * LPTIMx uint32_t Interrupt: { LL_LPTIM_IER_CMPMIE, LL_LPTIM_IER_CMPOKIE, LL_LPTIM_IER_ARRMIE, LL_LPTIM_IER_EXTTRIGIE, LL_LPTIM_IER_ARROKIE, LL_LPTIM_IER_UPIE, LL_LPTIM_IER_DOWNIE, LL_LPTIM_IER_IN1IDLE, LL_LPTIM_IER_IN2IDLE, LL_LPTIM_IER_INB2B, LL_LPTIM_IER_DUALERROR }
输出	无
返回值	无
资源使用	
说明	

2.18.120 LL_LPTIM_DisableIT

2.18.120.1 功能介绍

禁止使能比中断。

2.18.120.2 接口定义

函数接口	void LL_LPTIM_DisableIT (LPTIM_TypeDef * LPTIMx, uint32_t Interrupt)
输入	LPTIM_TypeDef * LPTIMx uint32_t Interrupt: { LL_LPTIM_IER_CMPMIE, LL_LPTIM_IER_CMPOKIE, LL_LPTIM_IER_ARRMIE, LL_LPTIM_IER_EXTTRIGIE, LL_LPTIM_IER_ARROKIE, LL_LPTIM_IER_UPIE, LL_LPTIM_IER_DOWNIE, LL_LPTIM_IER_IN1IDLE, LL_LPTIM_IER_IN2IDLE, LL_LPTIM_IER_INB2B, LL_LPTIM_IER_DUALERROR }
输出	无

返回值	无
资源使用	
说明	

2.18.121 LL_LPTIM_IsEnabledIT

2.18.121.1 功能介绍

检查是否使能中断。

2.18.121.2 接口定义

函数接口	uint32_t LL_LPTIM_IsEnabledIT (LPTIM_TypeDef * LPTIMx, uint32_t Interrupt)
输入	LPTIM_TypeDef * LPTIMx uint32_t Interrupt: { LL_LPTIM_IER_CMPMIE, LL_LPTIM_IER_CMPOKIE, LL_LPTIM_IER_ARRMIE, LL_LPTIM_IER_EXTTRIGIE, LL_LPTIM_IER_ARROKIE, LL_LPTIM_IER_UPIE, LL_LPTIM_IER_DOWNIE, LL_LPTIM_IER_IN1IDLE, LL_LPTIM_IER_IN2IDLE, LL_LPTIM_IER_INB2B, LL_LPTIM_IER_DUALERROR }
输出	无
返回值	{0,1}
资源使用	
说明	

2.19 LPUART 模块

属性	类型	字段名	含义
USART_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	CR3	控制寄存器 3
读写	uint32_t	BRR	波特率分频寄存器
读写	uint32_t	GTPR	保护时间和预分频器寄存器
读写	uint32_t	RTOR	超时及块传输长度寄存器
只写	uint32_t	RQR	请求寄存器
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断标志清零寄存器
只读	uint32_t	RDR	接收数据寄存器

读写	uint32_t	TDR	发送数据寄存器
读写	uint32_t	PRESC	预分频器寄存器
LL_LPUART_InitTypeDef			
读写	uint32_t	PrescalerValue	波特率分频值
读写	uint32_t	BaudRate	波特率
读写	uint32_t	DataWidth	字符长度
读写	uint32_t	StopBits	停止位
读写	uint32_t	Parity	RS485 收发器使能 信号极性
读写	uint32_t	TransferDirection	传输方向
读写	uint32_t	HardwareFlowControl	硬件流控制

2.19.1 LL_LPUART_Enable

2.19.1.1 功能介绍

使能 LPUART。

2.19.1.2 接口定义

函数接口	void LL_LPUART_Enable (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.2 LL_LPUART_Disable

2.19.2.1 功能介绍

禁止使能 LPUART。

2.19.2.2 接口定义

函数接口	void LL_LPUART_Disable (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.3 LL_LPUART_IsEnabled

2.19.3.1 功能介绍

检查是否使能 LPUART。

2.19.3.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabled (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.4 LL_LPUART_EnableFIFO

2.19.4.1 功能介绍

使能 LPUART FIFO。

2.19.4.2 接口定义

函数接口	void LL_LPUART_EnableFIFO (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.5 LL_LPUART_DisableFIFO

2.19.5.1 功能介绍

禁止使能 LPUART FIFO。

2.19.5.2 接口定义

函数接口	void LL_LPUART_DisableFIFO (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.6 LL_LPUART_IsEnabledFIFO

2.19.6.1 功能介绍

检查是否使能 LPUART FIFO。

2.19.6.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledFIFO (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.7 LL_LPUART_SetTXFIFOThreshold

2.19.7.1 功能介绍

设置 LPUART 发送 FIFO 阈值。

2.19.7.2 接口定义

函数接口	void LL_LPUART_SetTXFIFOThreshold (USART_TypeDef * LPUARTx, uint32_t Threshold)
输入	USART_TypeDef * LPUARTx uint32_t Threshold: { LL_LPUART_FIFOTHRESHOLD_1_8, LL_LPUART_FIFOTHRESHOLD_1_4, LL_LPUART_FIFOTHRESHOLD_1_2, LL_LPUART_FIFOTHRESHOLD_3_4, LL_LPUART_FIFOTHRESHOLD_7_8, LL_LPUART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	无
资源使用	
说明	

2.19.8 LL_LPUART_GetTXFIFOThreshold

2.19.8.1 功能介绍

获取 LPUART 发送 FIFO 阈值。

2.19.8.2 接口定义

函数接口	uint32_t LL_LPUART_GetTXFIFOThreshold (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_FIFOTHRESHOLD_1_8, LL_LPUART_FIFOTHRESHOLD_1_4, LL_LPUART_FIFOTHRESHOLD_1_2, LL_LPUART_FIFOTHRESHOLD_3_4, LL_LPUART_FIFOTHRESHOLD_7_8, LL_LPUART_FIFOTHRESHOLD_8_8 }
资源使用	
说明	

2.19.9 LL_LPUART_SetRXFIFOThreshold

2.19.9.1 功能介绍

设置 LPUART 接收 FIFO 阈值。

2.19.9.2 接口定义

函数接口	void LL_LPUART_SetRXFIFOThreshold (USART_TypeDef * LPUARTx, uint32_t Threshold)
------	---

输入	USART_TypeDef * LPUARTx uint32_t Threshold: { LL_LPUART_FIFOTHRESHOLD_1_8, LL_LPUART_FIFOTHRESHOLD_1_4, LL_LPUART_FIFOTHRESHOLD_1_2, LL_LPUART_FIFOTHRESHOLD_3_4, LL_LPUART_FIFOTHRESHOLD_7_8, LL_LPUART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	无
资源使用	
说明	

2.19.10 LL_LPUART_GetRXFIFOThreshold

2.19.10.1 功能介绍

获取 LPUART 接收 FIFO 阈值。

2.19.10.2 接口定义

函数接口	uint32_t LL_LPUART_GetRXFIFOThreshold (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_FIFOTHRESHOLD_1_8, LL_LPUART_FIFOTHRESHOLD_1_4, LL_LPUART_FIFOTHRESHOLD_1_2, LL_LPUART_FIFOTHRESHOLD_3_4, LL_LPUART_FIFOTHRESHOLD_7_8, LL_LPUART_FIFOTHRESHOLD_8_8 }
资源使用	
说明	

2.19.11 LL_LPUART_ConfigFIFOsThreshold

2.19.11.1 功能介绍

配置 LPUART 发送和接收 FIFO 阈值。

2.19.11.2 接口定义

函数接口	void LL_LPUART_ConfigFIFOsThreshold (USART_TypeDef * LPUARTx, uint32_t TXThreshold, uint32_t RXThreshold)
输入	USART_TypeDef * LPUARTx uint32_t TXThreshold: { LL_LPUART_FIFOTHRESHOLD_1_8, LL_LPUART_FIFOTHRESHOLD_1_4, LL_LPUART_FIFOTHRESHOLD_1_2, LL_LPUART_FIFOTHRESHOLD_3_4, LL_LPUART_FIFOTHRESHOLD_7_8, LL_LPUART_FIFOTHRESHOLD_8_8 }

	uint32_t RXThreshold: { LL_LPUART_FIFOTHRESHOLD_1_8, LL_LPUART_FIFOTHRESHOLD_1_4, LL_LPUART_FIFOTHRESHOLD_1_2, LL_LPUART_FIFOTHRESHOLD_3_4, LL_LPUART_FIFOTHRESHOLD_7_8, LL_LPUART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	
资源使用	
说明	

2.19.12 LL_LPUART_EnableInStopMode

2.19.12.1 功能介绍

使能 LPUART 将 MCU 从低功耗模式唤醒。

2.19.12.2 接口定义

函数接口	void LL_LPUART_EnableInStopMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.13 LL_LPUART_DisableInStopMode

2.19.13.1 功能介绍

禁止使能 LPUART 将 MCU 从低功耗模式唤醒。

2.19.13.2 接口定义

函数接口	void LL_LPUART_DisableInStopMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.14 LL_LPUART_IsEnabledInStopMode

2.19.14.1 功能介绍

检查是否使能 LPUART 将 MCU 从低功耗模式唤醒。

2.19.14.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledInStopMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无

返回值	{0,1}
资源使用	
说明	

2.19.15LL_LPUART_EnableDirectionRx

2.19.15.1 功能介绍

使能 LPUART 接收器使能。

2.19.15.2 接口定义

函数接口	void LL_LPUART_EnableDirectionRx (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.16LL_LPUART_DisableDirectionRx

2.19.16.1 功能介绍

禁止使能 LPUART 接收器使能。

2.19.16.2 接口定义

函数接口	void LL_LPUART_DisableDirectionRx (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.17LL_LPUART_EnableDirectionTx

2.19.17.1 功能介绍

使能 LPUART 发送器使能。

2.19.17.2 接口定义

函数接口	void LL_LPUART_EnableDirectionTx (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.18LL_LPUART_DisableDirectionTx

2.19.18.1 功能介绍

禁止使能 LPUART 发送器使能。

2.19.18.2 接口定义

函数接口	void LL_LPUART_DisableDirectionTx (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.19LL_LPUART_SetTransferDirection

2.19.19.1 功能介绍

配置 LPUART 发送器、接收器的使能或禁止。

2.19.19.2 接口定义

函数接口	void LL_LPUART_SetTransferDirection (USART_TypeDef * LPUARTx, uint32_t TransferDirection)
输入	USART_TypeDef * LPUARTx uint32_t TransferDirection: { LL_LPUART_DIRECTION_NONE, LL_LPUART_DIRECTION_RX, LL_LPUART_DIRECTION_TX, LL_LPUART_DIRECTION_TX_RX }
输出	无
返回值	无
资源使用	
说明	

2.19.20LL_LPUART_GetTransferDirection

2.19.20.1 功能介绍

获取 LPUART 发送器、接收器的使能或禁止。

2.19.20.2 接口定义

函数接口	uint32_t LL_LPUART_GetTransferDirection (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx uint32_t TransferDirection: { LL_LPUART_DIRECTION_NONE, LL_LPUART_DIRECTION_RX, LL_LPUART_DIRECTION_TX, LL_LPUART_DIRECTION_TX_RX }
输出	无
返回值	无
资源使用	
说明	

2.19.21LL_LPUART_SetParity

2.19.21.1 功能介绍

配置 LPUART 奇偶校验。

2.19.21.2 接口定义

函数接口	void LL_LPUART_SetParity (USART_TypeDef * LPUARTx, uint32_t Parity)
输入	USART_TypeDef * LPUARTx uint32_t Parity: { LL_LPUART_PARITY_NONE, LL_LPUART_PARITY_EVEN, LL_LPUART_PARITY_ODD }
输出	无
返回值	无
资源使用	
说明	

2.19.22LL_LPUART_GetParity

2.19.22.1 功能介绍

获取 LPUART 奇偶校验。

2.19.22.2 接口定义

函数接口	uint32_t LL_LPUART_GetParity (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_PARITY_NONE, LL_LPUART_PARITY_EVEN, LL_LPUART_PARITY_ODD }
资源使用	
说明	

2.19.23LL_LPUART_SetWakeUpMethod

2.19.23.1 功能介绍

配置 LPUART 唤醒方式。

2.19.23.2 接口定义

函数接口	void LL_LPUART_SetWakeUpMethod (USART_TypeDef * LPUARTx, uint32_t Method)
输入	USART_TypeDef * LPUARTx uint32_t Method: { LL_LPUART_WAKEUP_IDLELINE, LL_LPUART_WAKEUP_ADDRESSMARK }
输出	无
返回值	无
资源使用	
说明	

2.19.24LL_LPUART_GetWakeUpMethod

2.19.24.1 功能介绍

获取 LPUART 唤醒方式。

2.19.24.2 接口定义

函数接口	uint32_t LL_LPUART_GetWakeUpMethod (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_WAKEUP_IDLELINE, LL_LPUART_WAKEUP_ADDRESSMARK }
资源使用	
说明	

2.19.25 LL_LPUART_SetDataWidth

2.19.25.1 功能介绍

配置 LPUART 字符长度。

2.19.25.2 接口定义

函数接口	void LL_LPUART_SetDataWidth (USART_TypeDef * LPUARTx, uint32_t DataWidth)
输入	USART_TypeDef * LPUARTx uint32_t DataWidth: { LL_LPUART_DATAWIDTH_7B, LL_LPUART_DATAWIDTH_8B, LL_LPUART_DATAWIDTH_9B }
输出	无
返回值	无
资源使用	
说明	

2.19.26 LL_LPUART_GetDataWidth

2.19.26.1 功能介绍

获取 LPUART 字符长度。

2.19.26.2 接口定义

函数接口	uint32_t LL_LPUART_GetDataWidth (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_DATAWIDTH_7B, LL_LPUART_DATAWIDTH_8B, LL_LPUART_DATAWIDTH_9B }
资源使用	
说明	

2.19.27 LL_LPUART_EnableMuteMode

2.19.27.1 功能介绍

使能静默模式。

2.19.27.2 接口定义

函数接口	void LL_LPUART_EnableMuteMode (USART_TypeDef * LPUARTx)
------	---

输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.28 LL_LPUART_DisableMuteMode

2.19.28.1 功能介绍

禁止使能静默模式。

2.19.28.2 接口定义

函数接口	void LL_LPUART_DisableMuteMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.29 LL_LPUART_IsEnabledMuteMode

2.19.29.1 功能介绍

检查是否使能静默模式。

2.19.29.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledMuteMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.30 LL_LPUART_SetPrescaler

2.19.30.1 功能介绍

设置时钟预分频。

2.19.30.2 接口定义

函数接口	void LL_LPUART_SetPrescaler (USART_TypeDef * LPUARTx, uint32_t PrescalerValue)
输入	USART_TypeDef * LPUARTx uint32_t PrescalerValue: { LL_LPUART_PRESCALER_DIV1, LL_LPUART_PRESCALER_DIV2, LL_LPUART_PRESCALER_DIV4, LL_LPUART_PRESCALER_DIV6, LL_LPUART_PRESCALER_DIV8, LL_LPUART_PRESCALER_DIV10, LL_LPUART_PRESCALER_DIV12,

	LL_LPUART_PRESCALER_DIV16, LL_LPUART_PRESCALER_DIV32, LL_LPUART_PRESCALER_DIV64, LL_LPUART_PRESCALER_DIV128, LL_LPUART_PRESCALER_DIV256 }
输出	无
返回值	无
资源使用	
说明	

2.19.31 LL_LPUART_GetPrescaler

2.19.31.1 功能介绍

获取时钟预分频。

2.19.31.2 接口定义

函数接口	uint32_t LL_LPUART_GetPrescaler (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_PRESCALER_DIV1, LL_LPUART_PRESCALER_DIV2, LL_LPUART_PRESCALER_DIV4, LL_LPUART_PRESCALER_DIV6, LL_LPUART_PRESCALER_DIV8, LL_LPUART_PRESCALER_DIV10, LL_LPUART_PRESCALER_DIV12, LL_LPUART_PRESCALER_DIV16, LL_LPUART_PRESCALER_DIV32, LL_LPUART_PRESCALER_DIV64, LL_LPUART_PRESCALER_DIV128, LL_LPUART_PRESCALER_DIV256 }
资源使用	
说明	

2.19.32 LL_LPUART_SetStopBitsLength

2.19.32.1 功能介绍

设置停止位长度。

2.19.32.2 接口定义

函数接口	void LL_LPUART_SetStopBitsLength (USART_TypeDef * LPUARTx, uint32_t StopBits)
输入	USART_TypeDef * LPUARTx uint32_t StopBits: { LL_LPUART_STOPBITS_1, LL_LPUART_STOPBITS_2 }
输出	无
返回值	无
资源使用	
说明	

2.19.33 LL_LPUART_GetStopBitsLength

2.19.33.1 功能介绍

获取停止位长度。

2.19.33.2 接口定义

函数接口	uint32_t LL_LPUART_GetStopBitsLength (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_STOPBITS_1, LL_LPUART_STOPBITS_2 }
资源使用	
说明	

2.19.34 LL_LPUART_ConfigCharacter

2.19.34.1 功能介绍

配置字符帧格式(数据宽度, 奇偶校验控制, 停止位)。

2.19.34.2 接口定义

函数接口	void LL_LPUART_ConfigCharacter (USART_TypeDef * LPUARTx, uint32_t DataWidth, uint32_t Parity, uint32_t StopBits)
输入	USART_TypeDef * LPUARTx uint32_t DataWidth: { LL_LPUART_DATAWIDTH_7B, LL_LPUART_DATAWIDTH_8B, LL_LPUART_DATAWIDTH_9B } uint32_t Parity: { LL_LPUART_PARITY_NONE, LL_LPUART_PARITY_EVEN, LL_LPUART_PARITY_ODD } uint32_t StopBits: { LL_LPUART_STOPBITS_1, LL_LPUART_STOPBITS_2 }
输出	无
返回值	无
资源使用	
说明	

2.19.35 LL_LPUART_SetTXRXSwap

2.19.35.1 功能介绍

设置 LPUART TX/RX 引脚交换。

2.19.35.2 接口定义

函数接口	void LL_LPUART_SetTXRXSwap (USART_TypeDef * LPUARTx, uint32_t SwapConfig)
输入	USART_TypeDef * LPUARTx uint32_t SwapConfig: { LL_LPUART_TXRX_STANDARD, LL_LPUART_TXRX_SWAPPED }

输出	无
返回值	无
资源使用	
说明	

2.19.36 LL_LPUART_GetTXRXSwap

2.19.36.1 功能介绍

获取 LPUART TX/RX 引脚交换。

2.19.36.2 接口定义

函数接口	uint32_t LL_LPUART_GetTXRXSwap (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_TXRX_STANDARD, LL_LPUART_TXRX_SWAPPED }
资源使用	
说明	

2.19.37 LL_LPUART_SetRXPinLevel

2.19.37.1 功能介绍

设置 LPUART RX 引脚有效电平反向。

2.19.37.2 接口定义

函数接口	void LL_LPUART_SetRXPinLevel (USART_TypeDef * LPUARTx, uint32_t PinInvMethod)
输入	USART_TypeDef * LPUARTx uint32_t PinInvMethod: { LL_LPUART_RXPIN_LEVEL_STANDARD, LL_LPUART_RXPIN_LEVEL_INVERTED }
输出	无
返回值	无
资源使用	
说明	

2.19.38 LL_LPUART_GetRXPinLevel

2.19.38.1 功能介绍

获取 LPUART RX 引脚有效电平反向。

2.19.38.2 接口定义

函数接口	uint32_t LL_LPUART_GetRXPinLevel (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_RXPIN_LEVEL_STANDARD, LL_LPUART_RXPIN_LEVEL_INVERTED }
资源使用	

说明	
----	--

2.19.39LL_LPUART_SetTXPinLevel

2.19.39.1 功能介绍

设置 LPUART TX 引脚有效电平反向。

2.19.39.2 接口定义

函数接口	void LL_LPUART_SetTXPinLevel (USART_TypeDef * LPUARTx, uint32_t PinInvMethod)
输入	USART_TypeDef * LPUARTx uint32_t PinInvMethod: { LL_LPUART_TXPIN_LEVEL_STANDARD, LL_LPUART_TXPIN_LEVEL_INVERTED }
输出	无
返回值	无
资源使用	
说明	

2.19.40LL_LPUART_GetTXPinLevel

2.19.40.1 功能介绍

获取 LPUART TX 引脚有效电平反向。

2.19.40.2 接口定义

函数接口	uint32_t LL_LPUART_GetTXPinLevel (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_TXPIN_LEVEL_STANDARD, LL_LPUART_TXPIN_LEVEL_INVERTED }
资源使用	
说明	

2.19.41 LL_LPUART_SetBinaryDataLogic

2.19.41.1 功能介绍

设置 LPUART 二进制数据极性反向。

2.19.41.2 接口定义

函数接口	void LL_LPUART_SetBinaryDataLogic (USART_TypeDef * LPUARTx, uint32_t DataLogic)
输入	USART_TypeDef * LPUARTx uint32_t DataLogic: { LL_LPUART_BINARY_LOGIC_POSITIVE, LL_LPUART_BINARY_LOGIC_NEGATIVE }
输出	无
返回值	无
资源使用	

说明	
----	--

2.19.42 LL_LPUART_GetBinaryDataLogic

2.19.42.1 功能介绍

获取 LPUART 二进制数据极性反向。

2.19.42.2 接口定义

函数接口	uint32_t LL_LPUART_GetBinaryDataLogic (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_BINARY_LOGIC_POSITIVE, LL_LPUART_BINARY_LOGIC_NEGATIVE }
资源使用	
说明	

2.19.43 LL_LPUART_SetTransferBitOrder

2.19.43.1 功能介绍

设置 LPUART MSB 是否优先。

2.19.43.2 接口定义

函数接口	void LL_LPUART_SetTransferBitOrder (USART_TypeDef * LPUARTx, uint32_t BitOrder)
输入	USART_TypeDef * LPUARTx uint32_t BitOrder: { LL_LPUART_BITORDER_LSBFIRST, LL_LPUART_BITORDER_MSBFIRST }
输出	无
返回值	无
资源使用	
说明	

2.19.44 LL_LPUART_GetTransferBitOrder

2.19.44.1 功能介绍

获取 LPUART MSB 是否优先。

2.19.44.2 接口定义

函数接口	uint32_t LL_LPUART_GetTransferBitOrder (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_BITORDER_LSBFIRST, LL_LPUART_BITORDER_MSBFIRST }
资源使用	
说明	

2.19.45 LL_LPUART_ConfigNodeAddress

2.19.45.1 功能介绍

设置 LPUART 本地节点地址。

2.19.45.2 接口定义

函数接口	void LL_LPUART_ConfigNodeAddress (USART_TypeDef * LPUARTx, uint32_t AddressLen, uint32_t NodeAddress)
输入	USART_TypeDef * LPUARTx uint32_t AddressLen: { LL_LPUART_ADDRESS_DETECT_4B, LL_LPUART_ADDRESS_DETECT_7B } uint32_t NodeAddress: [0, 255]
输出	无
返回值	无
资源使用	
说明	

2.19.46 LL_LPUART_GetNodeAddress

2.19.46.1 功能介绍

获取 LPUART 本地节点。

2.19.46.2 接口定义

函数接口	uint32_t LL_LPUART_GetNodeAddress (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	[0, 255]
资源使用	
说明	

2.19.47 LL_LPUART_GetNodeAddressLen

2.19.47.1 功能介绍

获取 LPUART 本地地址长度。

2.19.47.2 接口定义

函数接口	uint32_t LL_LPUART_GetNodeAddressLen (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_ADDRESS_DETECT_4B, LL_LPUART_ADDRESS_DETECT_7B }
资源使用	
说明	

2.19.48 LL_LPUART_EnableRTSHWFlowCtrl

2.19.48.1 功能介绍

使能 LPUART RTS。

2.19.48.2 接口定义

函数接口	void LL_LPUART_EnableRTSHWFlowCtrl (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.49 LL_LPUART_DisableRTSHWFlowCtrl

2.19.49.1 功能介绍

禁止使能 LPUART RTS。

2.19.49.2 接口定义

函数接口	void LL_LPUART_DisableRTSHWFlowCtrl (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.50 LL_LPUART_EnableCTSHWFlowCtrl

2.19.50.1 功能介绍

使能 LPUART CTS。

2.19.50.2 接口定义

函数接口	void LL_LPUART_EnableCTSHWFlowCtrl (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.51 LL_LPUART_DisableCTSHWFlowCtrl

2.19.51.1 功能介绍

禁止使能 LPUART CTS。

2.19.51.2 接口定义

函数接口	void LL_LPUART_DisableCTSHWFlowCtrl (USART_TypeDef * LPUARTx)
------	---

	LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.52 LL_LPUART_SetHWFlowCtrl

2.19.52.1 功能介绍

配置硬件流控制模式(CTS 和 RTS)。

2.19.52.2 接口定义

函数接口	void LL_LPUART_SetHWFlowCtrl (USART_TypeDef * LPUARTx, uint32_t HardwareFlowControl)
输入	USART_TypeDef * LPUARTx uint32_t HardwareFlowControl: { LL_LPUART_HWCONTROL_NONE, LL_LPUART_HWCONTROL_RTS, LL_LPUART_HWCONTROL_CTS, LL_LPUART_HWCONTROL_RTS_CTS, }
输出	无
返回值	无
资源使用	
说明	

2.19.53 LL_LPUART_GetHWFlowCtrl

2.19.53.1 功能介绍

获取硬件流控制模式(CTS 和 RTS)。

2.19.53.2 接口定义

函数接口	uint32_t LL_LPUART_GetHWFlowCtrl (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_HWCONTROL_NONE, LL_LPUART_HWCONTROL_RTS, LL_LPUART_HWCONTROL_CTS, LL_LPUART_HWCONTROL_RTS_CTS, }
资源使用	
说明	

2.19.54 LL_LPUART_EnableOverrunDetect

2.19.54.1 功能介绍

使能上溢检测。

2.19.54.2 接口定义

函数接口	void LL_LPUART_EnableOverrunDetect (USART_TypeDef * LPUARTx)
------	--

输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.55 LL_LPUART_DisableOverrunDetect

2.19.55.1 功能介绍

禁止使能上溢检测。

2.19.55.2 接口定义

函数接口	void LL_LPUART_DisableOverrunDetect (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.56 LL_LPUART_IsEnabledOverrunDetect

2.19.56.1 功能介绍

检查是否使能上溢检测。

2.19.56.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledOverrunDetect (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.57 LL_LPUART_SetWKUPType

2.19.57.1 功能介绍

设置唤醒中断的事件类型。

2.19.57.2 接口定义

函数接口	void LL_LPUART_SetWKUPType (USART_TypeDef * LPUARTx, uint32_t Type)
输入	USART_TypeDef * LPUARTx uint32_t Type: { LL_LPUART_WAKEUP_ON_ADDRESS, LL_LPUART_WAKEUP_ON_STARTBIT, LL_LPUART_WAKEUP_ON_RXNE }
输出	无

返回值	无
资源使用	
说明	

2.19.58 LL_LPUART_GetWKUPTType

2.19.58.1 功能介绍

获取唤醒中断的事件类型。

2.19.58.2 接口定义

函数接口	uint32_t LL_LPUART_GetWKUPTType (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{ LL_LPUART_WAKEUP_ON_ADDRESS, LL_LPUART_WAKEUP_ON_STARTBIT, LL_LPUART_WAKEUP_ON_RXNE }
资源使用	
说明	

2.19.59 LL_LPUART_SetBaudRate

2.19.59.1 功能介绍

设置 LPUART 波特率。

2.19.59.2 接口定义

函数接口	void LL_LPUART_SetBaudRate (USART_TypeDef * LPUARTx, uint32_t PeriphClk, uint32_t PrescalerValue, uint32_t BaudRate)
输入	USART_TypeDef * LPUARTx uint32_t PeriphClk uint32_t PrescalerValue: { LL_LPUART_PRESCALER_DIV1, LL_LPUART_PRESCALER_DIV2, LL_LPUART_PRESCALER_DIV4, LL_LPUART_PRESCALER_DIV6, LL_LPUART_PRESCALER_DIV8, LL_LPUART_PRESCALER_DIV10 LL_LPUART_PRESCALER_DIV12, LL_LPUART_PRESCALER_DIV16, LL_LPUART_PRESCALER_DIV32, LL_LPUART_PRESCALER_DIV64, LL_LPUART_PRESCALER_DIV128, LL_LPUART_PRESCALER_DIV256 } uint32_t BaudRate
输出	无
返回值	无
资源使用	
说明	

2.19.60 LL_LPUART_GetBaudRate

2.19.60.1 功能介绍

获取 LPUART 波特率。

2.19.60.2 接口定义

函数接口	uint32_t LL_LPUART_GetBaudRate (USART_TypeDef * LPUARTx, uint32_t PeriphClk, uint32_t PrescalerValue)
输入	USART_TypeDef * LPUARTx uint32_t PeriphClk uint32_t PrescalerValue: { LL_LPUART_PRESCALER_DIV1, LL_LPUART_PRESCALER_DIV2, LL_LPUART_PRESCALER_DIV4, LL_LPUART_PRESCALER_DIV6, LL_LPUART_PRESCALER_DIV8, LL_LPUART_PRESCALER_DIV10 LL_LPUART_PRESCALER_DIV12, LL_LPUART_PRESCALER_DIV16, LL_LPUART_PRESCALER_DIV32, LL_LPUART_PRESCALER_DIV64, LL_LPUART_PRESCALER_DIV128, LL_LPUART_PRESCALER_DIV256 }
输出	无
返回值	波特率
资源使用	
说明	

2.19.61 LL_LPUART_EnableHalfDuplex

2.19.61.1 功能介绍

使能单线半双工模式。

2.19.61.2 接口定义

函数接口	void LL_LPUART_EnableHalfDuplex (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.62 LL_LPUART_DisableHalfDuplex

2.19.62.1 功能介绍

禁止使能单线半双工模式。

2.19.62.2 接口定义

函数接口	void LL_LPUART_DisableHalfDuplex (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无

返回值	无
资源使用	
说明	

2.19.63 LL_LPUART_IsEnabledHalfDuplex

2.19.63.1 功能介绍

检查是否使能单线半双工模式。

2.19.63.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledHalfDuplex (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.64 LL_LPUART_SetDEDeassertionTime

2.19.64.1 功能介绍

RS485 模式下，设置禁止驱动器时 DE 信号的保持时间。

2.19.64.2 接口定义

函数接口	void LL_LPUART_SetDEDeassertionTime (USART_TypeDef * LPUARTx, uint32_t Time)
输入	USART_TypeDef * LPUARTx uint32_t Time: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.19.65 LL_LPUART_GetDEDeassertionTime

2.19.65.1 功能介绍

RS485 模式下，获取禁止驱动器时 DE 信号的保持时间。

2.19.65.2 接口定义

函数接口	uint32_t LL_LPUART_GetDEDeassertionTime (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	[0, 31]
资源使用	
说明	

2.19.66 LL_LPUART_SetDEAssertionTime

2.19.66.1 功能介绍

RS485 模式下，设置 DE 信号的建立时间。

2.19.66.2 接口定义

函数接口	void LL_LPUART_SetDEAssertionTime (USART_TypeDef * LPUARTx, uint32_t Time)
输入	USART_TypeDef * LPUARTx uint32_t Time: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.19.67 LL_LPUART_GetDEAssertionTime

2.19.67.1 功能介绍

RS485 模式下，获取 DE 信号的建立时间。

2.19.67.2 接口定义

函数接口	uint32_t LL_LPUART_GetDEAssertionTime (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	[0, 31]
资源使用	
说明	

2.19.68 LL_LPUART_EnableDEMode

2.19.68.1 功能介绍

使能 DE 功能。

2.19.68.2 接口定义

函数接口	void LL_LPUART_EnableDEMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.69 LL_LPUART_DisableDEMode

2.19.69.1 功能介绍

禁止使能 DE 功能。

2.19.69.2 接口定义

函数接口	void LL_LPUART_DisableDEMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.70 LL_LPUART_IsEnabledDEMode

2.19.70.1 功能介绍

检查是否使能 DE 功能。

2.19.70.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledDEMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.71 LL_LPUART_SetDESignalPolarity

2.19.71.1 功能介绍

设置 RS485 收发器使能信号极性。

2.19.71.2 接口定义

函数接口	void LL_LPUART_SetDESignalPolarity (USART_TypeDef * LPUARTx, uint32_t Polarity)
输入	USART_TypeDef * LPUARTx uint32_t Polarity: { LL_LPUART_DE_POLARITY_HIGH, LL_LPUART_DE_POLARITY_LOW }
输出	无
返回值	无
资源使用	
说明	

2.19.72 LL_LPUART_GetDESignalPolarity

2.19.72.1 功能介绍

获取 RS485 收发器使能信号极性。

2.19.72.2 接口定义

函数接口	uint32_t LL_LPUART_GetDESignalPolarity (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx

输出	无
返回值	{ LL_LPUART_DE_POLARITY_HIGH, LL_LPUART_DE_POLARITY_LOW }
资源使用	
说明	

2.19.73 LL_LPUART_IsActiveFlag_PE

2.19.73.1 功能介绍

检查奇偶校验错误标志。

2.19.73.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_PE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.74 LL_LPUART_IsActiveFlag_FE

2.19.74.1 功能介绍

检查帧错误标志。

2.19.74.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_FE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.75 LL_LPUART_IsActiveFlag_NE

2.19.75.1 功能介绍

检查噪声检测标志。

2.19.75.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_NE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.76 LL_LPUART_IsActiveFlag_ORE

2.19.76.1 功能介绍

检查上溢错误标志。

2.19.76.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_ORE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.77 LL_LPUART_IsActiveFlag_IDLE

2.19.77.1 功能介绍

检查检测到空闲帧标志。

2.19.77.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_IDLE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.78 LL_LPUART_IsActiveFlag_RXNE_RXFNE

2.19.78.1 功能介绍

检查 RXFIFO 非空标志。

2.19.78.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_RXNE_RXFNE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.79 LL_LPUART_IsActiveFlag_TC

2.19.79.1 功能介绍

检查发送完成标志。

2.19.79.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_TC (USART_TypeDef * LPUARTx)
------	--

	LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.80 LL_LPUART_IsActiveFlag_TXE_TXFNF

2.19.80.1 功能介绍

检查 TXFIFO 未滿标志。

2.19.80.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_TXE_TXFNF (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.81 LL_LPUART_IsActiveFlag_nCTS

2.19.81.1 功能介绍

检查 CTS 中断标志。

2.19.81.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_nCTS (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.82 LL_LPUART_IsActiveFlag_CTS

2.19.82.1 功能介绍

检查 CTS 标志。

2.19.82.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_CTS (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	

说明	
----	--

2.19.83 LL_LPUART_IsActiveFlag_BUSY

2.19.83.1 功能介绍

检查忙标志。

2.19.83.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_BUSY (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.84 LL_LPUART_IsActiveFlag_CM

2.19.84.1 功能介绍

检查字符匹配标志。

2.19.84.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_CM (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.85 LL_LPUART_IsActiveFlag_SBK

2.19.85.1 功能介绍

检查中断帧发送标志。

2.19.85.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_SBK (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.86 LL_LPUART_IsActiveFlag_RWU

2.19.86.1 功能介绍

检查静默模式状态指示位。

2.19.86.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_RWU (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.87 LL_LPUART_IsActiveFlag_WKUP

2.19.87.1 功能介绍

检查从低功耗模式唤醒标志。

2.19.87.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_WKUP (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.88 LL_LPUART_IsActiveFlag_TEACK

2.19.88.1 功能介绍

检查发送使能确认标志。

2.19.88.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_TEACK (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.89 LL_LPUART_IsActiveFlag_REACK

2.19.89.1 功能介绍

检查接收使能确认标志。

2.19.89.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_REACK (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无

返回值	{0,1}
资源使用	
说明	

2.19.90 LL_LPUART_IsActiveFlag_TXFE

2.19.90.1 功能介绍

检查 TXFIFO 空标志。

2.19.90.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_TXFE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.91 LL_LPUART_IsActiveFlag_RXFE

2.19.91.1 功能介绍

检查 RXFIFO 已满标志。

2.19.91.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_RXFE(USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.92 LL_LPUART_IsActiveFlag_TXFT

2.19.92.1 功能介绍

检查 TXFIFO 阈值标志。

2.19.92.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_TXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.93 LL_LPUART_IsActiveFlag_RXFT

2.19.93.1 功能介绍

检查 RXFIFO 阈值标志。

2.19.93.2 接口定义

函数接口	uint32_t LL_LPUART_IsActiveFlag_RXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.94 LL_LPUART_ClearFlag_PE

2.19.94.1 功能介绍

清除奇偶校验错误标志。

2.19.94.2 接口定义

函数接口	void LL_LPUART_ClearFlag_PE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.95 LL_LPUART_ClearFlag_FE

2.19.95.1 功能介绍

清除帧错误标志。

2.19.95.2 接口定义

函数接口	void LL_LPUART_ClearFlag_FE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.96 LL_LPUART_ClearFlag_NE

2.19.96.1 功能介绍

清除噪声检测标志。

2.19.96.2 接口定义

函数接口	void LL_LPUART_ClearFlag_NE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx

输出	无
返回值	无
资源使用	
说明	

2.19.97 LL_LPUART_ClearFlag_ORE

2.19.97.1 功能介绍

清除上溢错误标志。

2.19.97.2 接口定义

函数接口	void LL_LPUART_ClearFlag_ORE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.98 LL_LPUART_ClearFlag_IDLE

2.19.98.1 功能介绍

清除空闲线路标志。

2.19.98.2 接口定义

函数接口	void LL_LPUART_ClearFlag_IDLE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.99 LL_LPUART_ClearFlag_TC

2.19.99.1 功能介绍

清除发送完成标志。

2.19.99.2 接口定义

函数接口	void LL_LPUART_ClearFlag_TC (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.100 LL_LPUART_ClearFlag_nCTS

2.19.100.1 功能介绍

清除 CTS 标志。

2.19.100.2 接口定义

函数接口	void LL_LPUART_ClearFlag_nCTS (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.101 LL_LPUART_ClearFlag_CM

2.19.101.1 功能介绍

清除字符匹配标志。

2.19.101.2 接口定义

函数接口	void LL_LPUART_ClearFlag_CM (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.102 LL_LPUART_ClearFlag_WKUP

2.19.102.1 功能介绍

清除从低功耗模式唤醒标志。

2.19.102.2 接口定义

函数接口	void LL_LPUART_ClearFlag_WKUP (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.103 LL_LPUART_EnableIT_IDLE

2.19.103.1 功能介绍

使能空闲帧检测中断。

2.19.103.2 接口定义

函数接口	void LL_LPUART_EnableIT_IDLE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.104 LL_LPUART_EnableIT_RXNE_RXFNE

2.19.104.1 功能介绍

使能接收数据非空中断。

2.19.104.2 接口定义

函数接口	void LL_LPUART_EnableIT_RXNE_RXFNE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.105 LL_LPUART_EnableIT_TC

2.19.105.1 功能介绍

使能传输完成中断。

2.19.105.2 接口定义

函数接口	void LL_LPUART_EnableIT_TC (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.106 LL_LPUART_EnableIT_TXE_TXFNF

2.19.106.1 功能介绍

使能数据寄存器空中断。

2.19.106.2 接口定义

函数接口	void LL_LPUART_EnableIT_TXE_TXFNF (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.107 LL_LPUART_EnableIT_PE

2.19.107.1 功能介绍

使能奇偶校验错误中断。

2.19.107.2 接口定义

函数接口	void LL_LPUART_EnableIT_PE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx

输出	无
返回值	无
资源使用	
说明	

2.19.108 LL_LPUART_EnableIT_CM

2.19.108.1 功能介绍

使能字符匹配中断。

2.19.108.2 接口定义

函数接口	void LL_LPUART_EnableIT_CM (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.109 LL_LPUART_EnableIT_TXFE

2.19.109.1 功能介绍

使能发送 FIFO 阈值中断。

2.19.109.2 接口定义

函数接口	void LL_LPUART_EnableIT_TXFE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.110 LL_LPUART_EnableIT_RXFF

2.19.110.1 功能介绍

使能接收 FIFO 阈值中断。

2.19.110.2 接口定义

函数接口	void LL_LPUART_EnableIT_RXFF (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.111 LL_LPUART_EnableIT_ERROR

2.19.111.1 功能介绍

使能错误中断。

2.19.111.2 接口定义

函数接口	void LL_LPUART_EnableIT_ERROR (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.112 LL_LPUART_EnableIT_CTS

2.19.112.1 功能介绍

使能 CTS 中断。

2.19.112.2 接口定义

函数接口	void LL_LPUART_EnableIT_CTS (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.113 LL_LPUART_EnableIT_WKUP

2.19.113.1 功能介绍

使能从低功耗模式唤醒信号中断。

2.19.113.2 接口定义

函数接口	void LL_LPUART_EnableIT_WKUP (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.114 LL_LPUART_EnableIT_TXFT

2.19.114.1 功能介绍

使能发送 FIFO 阈值中断。

2.19.114.2 接口定义

函数接口	void LL_LPUART_EnableIT_TXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.115 LL_LPUART_EnableIT_RXFT

2.19.115.1 功能介绍

使能接收 FIFO 阈值中断。

2.19.115.2 接口定义

函数接口	void LL_LPUART_EnableIT_RXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.116 LL_LPUART_DisableIT_IDLE

2.19.116.1 功能介绍

禁止使能空闲帧检测中断。

2.19.116.2 接口定义

函数接口	void LL_LPUART_DisableIT_IDLE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.117 LL_LPUART_DisableIT_RXNE_RXFNE

2.19.117.1 功能介绍

禁止使能接收数据非空中断。

2.19.117.2 接口定义

函数接口	void LL_LPUART_DisableIT_RXNE_RXFNE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.118 LL_LPUART_DisableIT_TC

2.19.118.1 功能介绍

禁止使能传输完成中断。

2.19.118.2 接口定义

函数接口	void LL_LPUART_DisableIT_TC (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx

输出	无
返回值	无
资源使用	
说明	

2.19.119 LL_LPUART_DisableIT_TXE_TXFNF

2.19.119.1 功能介绍

禁止使能数据寄存器空中断。

2.19.119.2 接口定义

函数接口	void LL_LPUART_DisableIT_TXE_TXFNF (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.120 LL_LPUART_DisableIT_PE

2.19.120.1 功能介绍

禁止使能奇偶校验错误中断。

2.19.120.2 接口定义

函数接口	void LL_LPUART_DisableIT_PE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.121 LL_LPUART_DisableIT_CM

2.19.121.1 功能介绍

禁止使能字符匹配中断。

2.19.121.2 接口定义

函数接口	void LL_LPUART_DisableIT_CM (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.122 LL_LPUART_DisableIT_TXFE

2.19.122.1 功能介绍

禁止使能发送 FIFO 阈值中断。

2.19.122.2 接口定义

函数接口	void LL_LPUART_DisableIT_TXFE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.123 LL_LPUART_DisableIT_RXFF

2.19.123.1 功能介绍

禁止使能接收 FIFO 阈值中断。

2.19.123.2 接口定义

函数接口	void LL_LPUART_DisableIT_RXFF (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.124 LL_LPUART_DisableIT_ERROR

2.19.124.1 功能介绍

禁止使能错误中断。

2.19.124.2 接口定义

函数接口	void LL_LPUART_DisableIT_ERROR (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.125 LL_LPUART_DisableIT_CTS

2.19.125.1 功能介绍

禁止使能 CTS 中断。

2.19.125.2 接口定义

函数接口	void LL_LPUART_DisableIT_CTS (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.126 LL_LPUART_DisableIT_WKUP

2.19.126.1 功能介绍

禁止使能从低功耗模式唤醒信号中断。

2.19.126.2 接口定义

函数接口	void LL_LPUART_DisableIT_WKUP (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.127 LL_LPUART_DisableIT_TXFT

2.19.127.1 功能介绍

禁止使能发送 FIFO 阈值中断。

2.19.127.2 接口定义

函数接口	void LL_LPUART_DisableIT_TXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.128 LL_LPUART_DisableIT_RXFT

2.19.128.1 功能介绍

禁止使能接收 FIFO 阈值中断。

2.19.128.2 接口定义

函数接口	void LL_LPUART_DisableIT_RXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.129 LL_LPUART_IsEnabledIT_IDLE

2.19.129.1 功能介绍

检查是否使能空闲帧检测中断。

2.19.129.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_IDLE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx

输出	无
返回值	{0,1}
资源使用	
说明	

2.19.130 LL_LPUART_IsEnabledIT_RXNE_RXFNE

2.19.130.1 功能介绍

检查是否使能接收数据非空中断。

2.19.130.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_RXNE_RXFNE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.131 LL_LPUART_IsEnabledIT_TC

2.19.131.1 功能介绍

检查是否使能传输完成中断。

2.19.131.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_TC (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.132 LL_LPUART_IsEnabledIT_TXE_TXFNF

2.19.132.1 功能介绍

检查是否使能数据寄存器空中断。

2.19.132.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_TXE_TXFNF (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.133 LL_LPUART_IsEnabledIT_PE

2.19.133.1 功能介绍

检查是否使能奇偶校验错误中断。

2.19.133.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_PE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.134 LL_LPUART_IsEnabledIT_CM

2.19.134.1 功能介绍

检查是否使能字符匹配中断。

2.19.134.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_CM (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.135 LL_LPUART_IsEnabledIT_TXFE

2.19.135.1 功能介绍

检查是否使能发送 FIFO 阈值中断。

2.19.135.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_TXFE (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.136 LL_LPUART_IsEnabledIT_RXFF

2.19.136.1 功能介绍

检查是否使能接收 FIFO 阈值中断。

2.19.136.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_RXFF (USART_TypeDef * LPUARTx)
------	---

输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.137 LL_LPUART_IsEnabledIT_ERROR

2.19.137.1 功能介绍

检查是否使能错误中断。

2.19.137.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_ERROR (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.138 LL_LPUART_IsEnabledIT_CTS

2.19.138.1 功能介绍

检查是否使能 CTS 中断。

2.19.138.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_CTS (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.139 LL_LPUART_IsEnabledIT_WKUP

2.19.139.1 功能介绍

检查是否使能从低功耗模式唤醒信号中断。

2.19.139.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_WKUP (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.140 LL_LPUART_IsEnabledIT_TXFT

2.19.140.1 功能介绍

检查是否使能发送 FIFO 阈值中断。

2.19.140.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_TXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.141 LL_LPUART_IsEnabledIT_RXFT

2.19.141.1 功能介绍

检查是否使能接收 FIFO 阈值中断。

2.19.141.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledIT_RXFT (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.142 LL_LPUART_EnableDMAReq_RX

2.19.142.1 功能介绍

使能 DMA 模式接收。

2.19.142.2 接口定义

函数接口	void LL_LPUART_EnableDMAReq_RX (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.143 LL_LPUART_DisableDMAReq_RX

2.19.143.1 功能介绍

禁止使能 DMA 模式接收。

2.19.143.2 接口定义

函数接口	void LL_LPUART_DisableDMAReq_RX (USART_TypeDef * LPUARTx)
------	---

	LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.144 LL_LPUART_IsEnabledDMAReq_RX

2.19.144.1 功能介绍

使能 DMA 模式接收。

2.19.144.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledDMAReq_RX (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.145 LL_LPUART_EnableDMAReq_TX

2.19.145.1 功能介绍

使能 DMA 模式发送。

2.19.145.2 接口定义

函数接口	void LL_LPUART_EnableDMAReq_RX (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.146 LL_LPUART_DisableDMAReq_TX

2.19.146.1 功能介绍

禁止使能 DMA 模式发送。

2.19.146.2 接口定义

函数接口	void LL_LPUART_DisableDMAReq_RX (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	

说明	
----	--

2.19.147 LL_LPUART_IsEnabledDMAReq_TX

2.19.147.1 功能介绍

使能 DMA 模式发送。

2.19.147.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledDMAReq_RX (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.19.148 LL_LPUART_EnableDMADeactOnRxErr

2.19.148.1 功能介绍

接收出错时禁止 DMA。

2.19.148.2 接口定义

函数接口	void LL_LPUART_EnableDMADeactOnRxErr (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.149 LL_LPUART_DisableDMADeactOnRxErr

2.19.149.1 功能介绍

接收出错时不禁止 DMA。

2.19.149.2 接口定义

函数接口	void LL_LPUART_DisableDMADeactOnRxErr (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.150 LL_LPUART_IsEnabledDMADeactOnRxErr

2.19.150.1 功能介绍

检查接收出错时是否禁止 DMA。

2.19.150.2 接口定义

函数接口	uint32_t LL_LPUART_IsEnabledDMADeactOnRxErr (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.151 LL_LPUART_DMA_GetRegAddr

2.19.151.1 功能介绍

获取用于 DMA 传输的 LPUART 数据寄存器地址。

2.19.151.2 接口定义

函数接口	uint32_t LL_LPUART_DMA_GetRegAddr (USART_TypeDef * LPUARTx, uint32_t Direction)
输入	USART_TypeDef * LPUARTx uint32_t Direction: { LL_LPUART_DMA_REG_DATA_TRANSMIT, LL_LPUART_DMA_REG_DATA_RECEIVE }
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.19.152 LL_LPUART_ReceiveData8

2.19.152.1 功能介绍

读取接收数据寄存器(接收数据值, 8 位)。

2.19.152.2 接口定义

函数接口	uint8_t LL_LPUART_ReceiveData8 (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.19.153 LL_LPUART_ReceiveData9

2.19.153.1 功能介绍

读取接收数据寄存器(接收数据值, 9 位)。

2.19.153.2 接口定义

函数接口	uint16_t LL_LPUART_ReceiveData9 (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx

输出	无
返回值	[0x000, 0x1FF]
资源使用	
说明	

2.19.154 LL_LPUART_TransmitData8

2.19.154.1 功能介绍

写入发送数据寄存器(发送数据值, 8 位)。

2.19.154.2 接口定义

函数接口	void LL_LPUART_TransmitData8 (USART_TypeDef * LPUARTx, uint8_t Value)
输入	USART_TypeDef * LPUARTx uint8_t Value: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.19.155 LL_LPUART_TransmitData9

2.19.155.1 功能介绍

写入发送数据寄存器(发送数据值, 9 位)。

2.19.155.2 接口定义

函数接口	void LL_LPUART_TransmitData9 (USART_TypeDef * LPUARTx, uint16_t Value)
输入	USART_TypeDef * LPUARTx uint16_t Value: [0x000, 0x1FF]
输出	无
返回值	无
资源使用	
说明	

2.19.156 LL_LPUART_RequestBreakSending

2.19.156.1 功能介绍

请求中断发送。

2.19.156.2 接口定义

函数接口	void LL_LPUART_RequestBreakSending (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无

资源使用	
说明	

2.19.157 LL_LPUART_RequestEnterMuteMode

2.19.157.1 功能介绍

将 LPUART 设置为静音模式，并设置 RWU 标志。

2.19.157.2 接口定义

函数接口	void LL_LPUART_RequestEnterMuteMode (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.158 LL_LPUART_RequestRxDataFlush

2.19.158.1 功能介绍

请求接收数据和 FIFO 刷新。

2.19.158.2 接口定义

函数接口	void LL_LPUART_RequestRxDataFlush (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.159 LL_LPUART_RequestTxDataFlush

2.19.159.1 功能介绍

请求发送数据和 FIFO 刷新。

2.19.159.2 接口定义

函数接口	void LL_LPUART_RequestTxDataFlush (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	无
资源使用	
说明	

2.19.160 LL_LPUART_DeInit

2.19.160.1 功能介绍

清除 LPUART 初始化参数。

2.19.160.2 接口定义

函数接口	ErrorStatus LL_LPUART_DeInit (USART_TypeDef * LPUARTx)
输入	USART_TypeDef * LPUARTx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.19.161 LL_LPUART_Init

2.19.161.1 功能介绍

LPUART 初始化。

2.19.161.2 接口定义

函数接口	ErrorStatus LL_LPUART_Init (USART_TypeDef * LPUARTx, LL_LPUART_InitTypeDef * LPUART_InitStruct)
输入	USART_TypeDef * LPUARTx LL_LPUART_InitTypeDef * LPUART_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.19.162 LL_LPUART_StructInit

2.19.162.1 功能介绍

LPUART 结构体初始化。

2.19.162.2 接口定义

函数接口	void LL_LPUART_StructInit (LL_LPUART_InitTypeDef * LPUART_InitStruct)
输入	LL_LPUART_InitTypeDef * LPUART_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.20 OPAMP 模块

属性	类型	字段名	含义
OPAMP_TypeDef			
读写	uint32_t	CSR	控制和状态寄存器
LL_OPAMP_InitTypeDef			
读写	uint32_t	InputNonInverting	正相输入
读写	uint32_t	InputInverting	反相输入

读写	uint32_t	PGAGain	输入增益
读写	uint32_t	Mode	模式
读写	uint32_t	OutputMode	输出模式
读写	uint32_t	TrimmingSource	校正源

2.20.1 LL_OPAMP_SetInputNonInverting

2.20.1.1 功能介绍

设置 OPAMP 正相输入连接。

2.20.1.2 接口定义

函数接口	void LL_OPAMP_SetInputNonInverting (OPAMP_TypeDef * OPAMPx, uint32_t InputNonInverting)
输入	OPAMP_TypeDef * OPAMPx uint32_t InputNonInverting: { LL_OPAMP_INPUT_NONINVERT_IO1, LL_OPAMP_INPUT_NONINVERT_IO2, LL_OPAMP_INPUT_NONINVERT_IO3, LL_OPAMP_INPUT_NONINVERT_IO4, LL_OPAMP_INPUT_NONINVERT_LOW }
输出	无
返回值	无
资源使用	
说明	

2.20.2 LL_OPAMP_GetInputNonInverting

2.20.2.1 功能介绍

获取 OPAMP 正相输入连接。

2.20.2.2 接口定义

函数接口	uint32_t LL_OPAMP_GetInputNonInverting (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{ LL_OPAMP_INPUT_NONINVERT_IO1, LL_OPAMP_INPUT_NONINVERT_IO2, LL_OPAMP_INPUT_NONINVERT_IO3, LL_OPAMP_INPUT_NONINVERT_IO4, LL_OPAMP_INPUT_NONINVERT_LOW }
资源使用	
说明	

2.20.3 LL_OPAMP_SetInputInverting

2.20.3.1 功能介绍

设置 OPAMP 反相输入连接。

2.20.3.2 接口定义

函数接口	void LL_OPAMP_SetInputInverting (OPAMP_TypeDef * OPAMPx, uint32_t InputInverting)
输入	OPAMP_TypeDef * OPAMPx uint32_t InputInverting: { LL_OPAMP_INPUT_INVERT_IO1, LL_OPAMP_INPUT_INVERT_IO2 }
输出	无
返回值	无
资源使用	
说明	

2.20.4 LL_OPAMP_GetInputInverting

2.20.4.1 功能介绍

获取 OPAMP 反相输入连接。

2.20.4.2 接口定义

函数接口	uint32_t LL_OPAMP_GetInputInverting (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{ LL_OPAMP_INPUT_INVERT_IO1, LL_OPAMP_INPUT_INVERT_IO2 }
资源使用	
说明	

2.20.5 LL_OPAMP_SetPGAGain

2.20.5.1 功能介绍

设置 OPAMP 输入增益。

2.20.5.2 接口定义

函数接口	void LL_OPAMP_SetPGAGain (OPAMP_TypeDef * OPAMPx, uint32_t PGAGain)
输入	OPAMP_TypeDef * OPAMPx uint32_t PGAGain: { LL_OPAMP_PGA_GAIN_20, LL_OPAMP_PGA_GAIN_25, LL_OPAMP_PGA_GAIN_30, LL_OPAMP_PGA_GAIN_35, LL_OPAMP_PGA_GAIN_100, LL_OPAMP_PGA_GAIN_105, LL_OPAMP_PGA_GAIN_110, LL_OPAMP_PGA_GAIN_115 }
输出	无
返回值	无
资源使用	
说明	

2.20.6 LL_OPAMP_GetPGAGain

2.20.6.1 功能介绍

获取 OPAMP 输入增益。

2.20.6.2 接口定义

函数接口	uint32_t LL_OPAMP_GetPGAGain (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{ LL_OPAMP_PGA_GAIN_20, LL_OPAMP_PGA_GAIN_25, LL_OPAMP_PGA_GAIN_30, LL_OPAMP_PGA_GAIN_35, LL_OPAMP_PGA_GAIN_100, LL_OPAMP_PGA_GAIN_105, LL_OPAMP_PGA_GAIN_110, LL_OPAMP_PGA_GAIN_115 }
资源使用	
说明	

2.20.7 LL_OPAMP_SetFunctionalMode

2.20.7.1 功能介绍

设置 OPAMP 操作模式。

2.20.7.2 接口定义

函数接口	void LL_OPAMP_SetFunctionalMode (OPAMP_TypeDef * OPAMPx, uint32_t FunctionalMode)
输入	OPAMP_TypeDef * OPAMPx uint32_t FunctionalMode: { LL_OPAMP_MODE_TRIMMING1, LL_OPAMP_MODE_TRIMMING2, LL_OPAMP_MODE_INPUT_INVERT, LL_OPAMP_MODE_INPUT_NONINVERT, LL_OPAMP_MODE_INPUT_INVERT_1KNONINVERT }
输出	无
返回值	无
资源使用	
说明	

2.20.8 LL_OPAMP_GetFunctionalMode

2.20.8.1 功能介绍

获取 OPAMP 操作模式。

2.20.8.2 接口定义

函数接口	uint32_t LL_OPAMP_GetFunctionalMode (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{ LL_OPAMP_MODE_TRIMMING1, LL_OPAMP_MODE_TRIMMING2,

	LL_OPAMP_MODE_INPUT_INVERT, LL_OPAMP_MODE_INPUT_NONINVERT, LL_OPAMP_MODE_INPUT_INVERT_1KNONINVERT }
资源使用	
说明	

2.20.9 LL_OPAMP_SetTrimmingSource

2.20.9.1 功能介绍

设置 OPAMP 校正源。

2.20.9.2 接口定义

函数接口	void LL_OPAMP_SetTrimmingSource (OPAMP_TypeDef * OPAMPx, uint32_t TrimmingSource)
输入	OPAMP_TypeDef * OPAMPx uint32_t TrimmingSource: { LL_OPAMP_TRIMMINGSOURCE_VSSA, LL_OPAMP_TRIMMINGSOURCE_VBGR }
输出	无
返回值	无
资源使用	
说明	

2.20.10 LL_OPAMP_GetTrimmingSource

2.20.10.1 功能介绍

获取 OPAMP 校正源。

2.20.10.2 接口定义

函数接口	uint32_t LL_OPAMP_GetTrimmingSource (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{ LL_OPAMP_TRIMMINGSOURCE_VSSA, LL_OPAMP_TRIMMINGSOURCE_VBGR }
资源使用	
说明	

2.20.11 LL_OPAMP_SetOutputMode

2.20.11.1 功能介绍

设置 OPAMP 校正源。

2.20.11.2 接口定义

函数接口	void LL_OPAMP_SetOutputMode (OPAMP_TypeDef * OPAMPx, uint32_t Mode)
输入	OPAMP_TypeDef * OPAMPx uint32_t Mode: { LL_OPAMP_OUTPUTTOCOMP_ENABLE,

	LL_OPAMP_OUTPUTTOCOMP_DISABLE }
输出	无
返回值	无
资源使用	
说明	

2.20.12 LL_OPAMP_GetOutputMode

2.20.12.1 功能介绍

获取 OPAMP 校正源。

2.20.12.2 接口定义

函数接口	uint32_t LL_OPAMP_GetOutputMode (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{ LL_OPAMP_OUTPUTTOCOMP_ENABLE, LL_OPAMP_OUTPUTTOCOMP_DISABLE }
资源使用	
说明	

2.20.13 LL_OPAMP_Enable

2.20.13.1 功能介绍

使能 OPAMP。

2.20.13.2 接口定义

函数接口	void LL_OPAMP_Enable (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	无
资源使用	
说明	

2.20.14 LL_OPAMP_Disable

2.20.14.1 功能介绍

禁止使能 OPAMP。

2.20.14.2 接口定义

函数接口	void LL_OPAMP_Disable (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	无
资源使用	
说明	

2.20.15 LL_OPAMP_IsEnabled

2.20.15.1 功能介绍

检查是否使能 OPAMP。

2.20.15.2 接口定义

函数接口	uint32_t LL_OPAMP_IsEnabled (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{0,1}
资源使用	
说明	

2.20.16 LL_OPAMP_Lock

2.20.16.1 功能介绍

锁定 OPAMP。

2.20.16.2 接口定义

函数接口	void LL_OPAMP_Lock (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	无
资源使用	
说明	

2.20.17 LL_OPAMP_IsLocked

2.20.17.1 功能介绍

检查是否锁定 OPAMP。

2.20.17.2 接口定义

函数接口	uint32_t LL_OPAMP_IsLocked (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	{0,1}
资源使用	
说明	

2.20.18 LL_OPAMP_DeInit

2.20.18.1 功能介绍

清除 OPAMP 初始化参数。

2.20.18.2 接口定义

函数接口	ErrorStatus LL_OPAMP_DeInit (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无

返回值	ErrorStatus
资源使用	
说明	

2.20.19LL_OPAMP_Init

2.20.19.1 功能介绍

OPAMP 初始化。

2.20.19.2 接口定义

函数接口	ErrorStatus LL_OPAMP_Init (OPAMP_TypeDef * OPAMPx)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.20.20LL_OPAMP_StructInit

2.20.20.1 功能介绍

OPAMP 结构体初始化。

2.20.20.2 接口定义

函数接口	void LL_OPAMP_StructInit (LL_OPAMP_InitTypeDef * OPAMP_InitStruct)
输入	OPAMP_TypeDef * OPAMPx
输出	无
返回值	无
资源使用	
说明	

2.21 PLA 模块

属性	类型	字段名	含义
PLA_TypeDef			
读写	uint32_t	CR	控制寄存器
读写	uint32_t	SR	标志寄存器
读写	uint32_t	CFGR0	PLU0 配置寄存器
读写	uint32_t	CFGR1	PLU1 配置寄存器
读写	uint32_t	CFGR2	PLU2 配置寄存器
读写	uint32_t	CFGR3	PLU3 配置寄存器

2.21.1 LL_PLA_SetMUXAInput

2.21.1.1 功能介绍

设置 PLA MUXA 输入选择。

2.21.1.2 接口定义

函数接口	void LL_PLA_SetMUXAInput (PLA_TypeDef * PLAx, uint32_t Port, uint32_t InputSel)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 } uint32_t InputSel: { LL_PLA_MUXA_0, LL_PLA_MUXA_1, LL_PLA_MUXA_2, LL_PLA_MUXA_3, LL_PLA_MUXA_4, LL_PLA_MUXA_5, LL_PLA_MUXA_6, LL_PLA_MUXA_7, LL_PLA_MUXA_8, LL_PLA_MUXA_9, LL_PLA_MUXA_10, LL_PLA_MUXA_11, LL_PLA_MUXA_12, LL_PLA_MUXA_13, LL_PLA_MUXA_14, LL_PLA_MUXA_15 }
输出	无
返回值	无
资源使用	
说明	

2.21.2 LL_PLA_GetMUXAInput

2.21.2.1 功能介绍

获取 PLA MUXA 输入选择。

2.21.2.2 接口定义

函数接口	uint32_t LL_PLA_GetMUXAInput (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{ LL_PLA_MUXA_0, LL_PLA_MUXA_1, LL_PLA_MUXA_2, LL_PLA_MUXA_3, LL_PLA_MUXA_4, LL_PLA_MUXA_5, LL_PLA_MUXA_6, LL_PLA_MUXA_7, LL_PLA_MUXA_8, LL_PLA_MUXA_9, LL_PLA_MUXA_10, LL_PLA_MUXA_11, LL_PLA_MUXA_12, LL_PLA_MUXA_13, LL_PLA_MUXA_14, LL_PLA_MUXA_15 }
资源使用	
说明	

2.21.3 LL_PLA_SetMUXBInput

2.21.3.1 功能介绍

设置 PLA MUXB 输入选择。

2.21.3.2 接口定义

函数接口	void LL_PLA_SetMUXBInput (PLA_TypeDef * PLAx, uint32_t Port, uint32_t InputSel)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 } uint32_t InputSel: { LL_PLA_MUXB_0, LL_PLA_MUXB_1, LL_PLA_MUXB_2, LL_PLA_MUXB_3, LL_PLA_MUXB_4, LL_PLA_MUXB_5, LL_PLA_MUXB_6, LL_PLA_MUXB_7, LL_PLA_MUXB_8, LL_PLA_MUXB_9, LL_PLA_MUXB_10, LL_PLA_MUXB_11, LL_PLA_MUXB_12, LL_PLA_MUXB_13, LL_PLA_MUXB_14, LL_PLA_MUXB_15 }
输出	无
返回值	无
资源使用	
说明	

2.21.4 LL_PLA_GetMUXBInput

2.21.4.1 功能介绍

获取 PLA MUXB 输入选择。

2.21.4.2 接口定义

函数接口	uint32_t LL_PLA_GetMUXBInput (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{ LL_PLA_MUXB_0, LL_PLA_MUXB_1, LL_PLA_MUXB_2, LL_PLA_MUXB_3, LL_PLA_MUXB_4, LL_PLA_MUXB_5, LL_PLA_MUXB_6, LL_PLA_MUXB_7, LL_PLA_MUXB_8, LL_PLA_MUXB_9, LL_PLA_MUXB_10, LL_PLA_MUXB_11, LL_PLA_MUXB_12, LL_PLA_MUXB_13, LL_PLA_MUXB_14, LL_PLA_MUXB_15 }
资源使用	
说明	

2.21.5 LL_PLA_EnableOutput

2.21.5.1 功能介绍

使能 PLA 输出。

2.21.5.2 接口定义

函数接口	void LL_PLA_EnableOutput (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.6 LL_PLA_DisableOutput

2.21.6.1 功能介绍

禁止使能 PLA 输出。

2.21.6.2 接口定义

函数接口	void LL_PLA_DisableOutput (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.7 LL_PLA_IsEnabledOutput

2.21.7.1 功能介绍

检查是否使能 PLA 输出。

2.21.7.2 接口定义

函数接口	uint32_t LL_PLA_IsEnabledOutput (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{0,1}
资源使用	

说明	
----	--

2.21.8 LL_PLA_SetFunctionMode

2.21.8.1 功能介绍

设置 PLA 功能模式。

2.21.8.2 接口定义

函数接口	void LL_PLA_SetFunctionMode (PLA_TypeDef * PLAx, uint32_t Port, uint32_t FuncSel)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 } uint32_t FuncSel: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.21.9 LL_PLA_GetFunctionMode

2.21.9.1 功能介绍

获取 PLA 功能模式。

2.21.9.2 接口定义

函数接口	uint32_t LL_PLA_GetFunctionMode (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	[0x00, 0x7F]
资源使用	
说明	

2.21.10 LL_PLA_SetOutputMode

2.21.10.1 功能介绍

设置 PLA 输出模式。

2.21.10.2 接口定义

函数接口	void LL_PLA_SetOutputMode (PLA_TypeDef * PLAx, uint32_t Port, uint32_t OutputSel)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2,

	LL_PLA_PORT_3 } uint32_t OutputSel: { LL_PLA_OUTPUT_DFF, LL_PLA_OUTPUT_LUT }
输出	无
返回值	无
资源使用	
说明	

2.21.11 LL_PLA_GetOutputMode

2.21.11.1 功能介绍

获取 PLA 输出模式。

2.21.11.2 接口定义

函数接口	uint32_t LL_PLA_GetOutputMode (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{ LL_PLA_OUTPUT_DFF, LL_PLA_OUTPUT_LUT }
资源使用	
说明	

2.21.12 LL_PLA_DFF_Reset

2.21.12.1 功能介绍

重置 PLA DFF。

2.21.12.2 接口定义

函数接口	void LL_PLA_DFF_Reset (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.13 LL_PLA_DFF_SetClockInverseMode

2.21.13.1 功能介绍

设置 PLA 时钟反相模式。

2.21.13.2 接口定义

函数接口	void LL_PLA_DFF_SetClockInverseMode (PLA_TypeDef * PLAx,
------	---

	uint32_t Port, uint32_t Mode)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 } uint32_t Mode: { LL_PLA_INPUTCLOCK_NONINVERSE, LL_PLA_INPUTCLOCK_INVERSE }
输出	无
返回值	无
资源使用	
说明	

2.21.14 LL_PLA_DFF_GetClockInverseMode

2.21.14.1 功能介绍

获取 PLA 时钟反相模式。

2.21.14.2 接口定义

函数接口	void LL_PLA_DFF_SetClockInverseMode (PLA_TypeDef * PLAx, uint32_t Port, uint32_t Mode)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{ LL_PLA_INPUTCLOCK_NONINVERSE, LL_PLA_INPUTCLOCK_INVERSE }
资源使用	
说明	

2.21.15 LL_PLA_DFF_SetClockSource

2.21.15.1 功能介绍

设置 PLA DFF 时钟源。

2.21.15.2 接口定义

函数接口	void LL_PLA_DFF_SetClockSource (PLA_TypeDef * PLAx, uint32_t Port, uint32_t ClockSource)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 } uint32_t ClockSource: { LL_PLA_INPUTCLOCK_CARRYIN, LL_PLA_INPUTCLOCK_MXA, LL_PLA_INPUTCLOCK_SYSCLK, LL_PLA_INPUTCLOCK_ALTCLK }

输出	无
返回值	无
资源使用	
说明	

2.21.16 LL_PLA_DFF_GetClockSource

2.21.16.1 功能介绍

获取 PLA DFF 时钟源。

2.21.16.2 接口定义

函数接口	uint32_t LL_PLA_DFF_GetClockSource (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{ LL_PLA_INPUTCLOCK_CARRYIN, LL_PLA_INPUTCLOCK_MXA, LL_PLA_INPUTCLOCK_SYSCLK, LL_PLA_INPUTCLOCK_ALTCLK }
资源使用	
说明	

2.21.17 LL_PLA_EnableIT_Rising

2.21.17.1 功能介绍

使能 PLA 上升沿中断。

2.21.17.2 接口定义

函数接口	void LL_PLA_EnableIT_Rising (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.18 LL_PLA_DisableIT_Rising

2.21.18.1 功能介绍

禁止使能 PLA 上升沿中断。

2.21.18.2 接口定义

函数接口	void LL_PLA_DisableIT_Rising (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx

	uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.19 LL_PLA_IsEnabledIT_Rising

2.21.19.1 功能介绍

检查是否使能 PLA 上升沿中断。

2.21.19.2 接口定义

函数接口	uint32_t LL_PLA_IsEnabledIT_Rising (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.21.20 LL_PLA_EnableIT_Falling

2.21.20.1 功能介绍

使能 PLA 下降沿中断。

2.21.20.2 接口定义

函数接口	void LL_PLA_EnableIT_Falling (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.21 LL_PLA_DisableIT_Falling

2.21.21.1 功能介绍

禁止使能 PLA 下降沿中断。

2.21.21.2 接口定义

函数接口	void LL_PLA_DisableIT_Falling (PLA_TypeDef * PLAx, uint32_t Port)
------	---

输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.22 LL_PLA_IsEnabledIT_Falling

2.21.22.1 功能介绍

检查是否使能 PLA 下降沿中断。

2.21.22.2 接口定义

函数接口	uint32_t LL_PLA_IsEnabledIT_Falling (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.21.23 LL_PLA_IsActiveFlag_Rising

2.21.23.1 功能介绍

获取 PLA 上升沿中断标志。

2.21.23.2 接口定义

函数接口	uint32_t LL_PLA_IsActiveFlag_Rising (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.21.24 LL_PLA_IsActiveFlag_Falling

2.21.24.1 功能介绍

获取 PLA 下降沿中断标志。

2.21.24.2 接口定义

函数接口	uint32_t LL_PLA_IsActiveFlag_Falling (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.21.25 LL_PLA_ClearFlag_Rising

2.21.25.1 功能介绍

清除 PLA 上升沿中断标志。

2.21.25.2 接口定义

函数接口	void LL_PLA_ClearFlag_Rising (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.26 LL_PLA_ClearFlag_Falling

2.21.26.1 功能介绍

清除 PLA 下降沿中断标志。

2.21.26.2 接口定义

函数接口	void LL_PLA_ClearFlag_Falling (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.27 LL_PLA_Enable

2.21.27.1 功能介绍

使能 PLA 。

2.21.27.2 接口定义

函数接口	void LL_PLA_Enable (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.28 LL_PLA_Disable

2.21.28.1 功能介绍

禁止使能 PLA 。

2.21.28.2 接口定义

函数接口	void LL_PLA_Disable (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	无
资源使用	
说明	

2.21.29 LL_PLA_IsEnabled

2.21.29.1 功能介绍

检查是否使能 PLA 。

2.21.29.2 接口定义

函数接口	uint32_t LL_PLA_IsEnabled (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.21.30 LL_PLA_ReadOutputLevel

2.21.30.1 功能介绍

获取 PLA 输出状态 。

2.21.30.2 接口定义

函数接口	uint32_t LL_PLA_ReadOutputLevel (PLA_TypeDef * PLAx, uint32_t Port)
输入	PLA_TypeDef * PLAx uint32_t Port: { LL_PLA_PORT_0, LL_PLA_PORT_1, LL_PLA_PORT_2, LL_PLA_PORT_3 }
输出	无
返回值	{ LL_PLA_OUTPUT_LEVEL_LOW, LL_PLA_OUTPUT_LEVEL_HIGH }
资源使用	
说明	

2.22 RCC 模块

属性	类型	字段名	含义
LL_RCC_ClocksTypeDef			
读写	uint32_t	SYSCLK_Frequency	SYSCLK 时钟频率
读写	uint32_t	HCLK_Frequency	HCLK 时钟频率
读写	uint32_t	PCLK1_Frequency	PCLK1 时钟频率
读写	uint32_t	PCLK2_Frequency	PCLK2 时钟频率

2.22.1 LL_RCC_HSE_EnableBypass

2.22.1.1 功能介绍

使能 HSE 旁路模式。

2.22.1.2 接口定义

函数接口	void LL_RCC_HSE_EnableBypass (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.2 LL_RCC_HSE_DisableBypass

2.22.2.1 功能介绍

禁止使能 HSE 旁路模式。

2.22.2.2 接口定义

函数接口	void LL_RCC_HSE_DisableBypass (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.22.3 LL_RCC_HSE_IsEnabledBypass

2.22.3.1 功能介绍

检查是否使能 HSE 旁路模式。

2.22.3.2 接口定义

函数接口	uint32_t LL_RCC_HSE_IsEnabledBypass (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.4 LL_RCC_HSE_Enable

2.22.4.1 功能介绍

使能 HSE。

2.22.4.2 接口定义

函数接口	void LL_RCC_HSE_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.5 LL_RCC_HSE_Disable

2.22.5.1 功能介绍

禁止使能 HSE。

2.22.5.2 接口定义

函数接口	void LL_RCC_HSE_Disable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.6 LL_RCC_HSE_IsEnabled

2.22.6.1 功能介绍

检查是否使能 HSE。

2.22.6.2 接口定义

函数接口	uint32_t LL_RCC_HSE_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.7 LL_RCC_HSE_SetState

2.22.7.1 功能介绍

设置 HSE 状态。

2.22.7.2 接口定义

函数接口	void LL_RCC_HSE_SetState (uint32_t HSEState)
输入	uint32_t HSEState: { LL_RCC_HSE_OFF, LL_RCC_HSE_ON, LL_RCC_HSE_BYPASS }
输出	无
返回值	无
资源使用	
说明	

2.22.8 LL_RCC_HSE_GetState

2.22.8.1 功能介绍

获取 HSE 状态。

2.22.8.2 接口定义

函数接口	uint32_t LL_RCC_HSE_GetState (void)
输入	无
输出	无
返回值	{ LL_RCC_HSE_OFF, LL_RCC_HSE_ON, LL_RCC_HSE_BYPASS }
资源使用	
说明	

2.22.9 LL_RCC_HSE_SetStableCycle

2.22.9.1 功能介绍

设置等待 HSE 时钟稳定的周期数。

2.22.9.2 接口定义

函数接口	void LL_RCC_HSE_SetStableCycle (uint32_t HSEStableCycle)
输入	uint32_t HSEStableCycle: { LL_RCC_HSE_STABLECYCLE_2, LL_RCC_HSE_STABLECYCLE_1024, LL_RCC_HSE_STABLECYCLE_4096, LL_RCC_HSE_STABLECYCLE_8192,

	LL_RCC_HSE_STABLECYCLE_16384, LL_RCC_HSE_STABLECYCLE_32768, LL_RCC_HSE_STABLECYCLE_65535, LL_RCC_HSE_STABLECYCLE_131072 }
输出	无
返回值	无
资源使用	
说明	

2.22.10 LL_RCC_HSE_GetStableCycle

2.22.10.1 功能介绍

获取等待 HSE 时钟稳定的周期数。

2.22.10.2 接口定义

函数接口	uint32_t LL_RCC_HSE_GetStableCycle (void)
输入	无
输出	无
返回值	{ LL_RCC_HSE_STABLECYCLE_2, LL_RCC_HSE_STABLECYCLE_1024, LL_RCC_HSE_STABLECYCLE_4096, LL_RCC_HSE_STABLECYCLE_8192, LL_RCC_HSE_STABLECYCLE_16384, LL_RCC_HSE_STABLECYCLE_32768, LL_RCC_HSE_STABLECYCLE_65535, LL_RCC_HSE_STABLECYCLE_131072 }
资源使用	
说明	

2.22.11 LL_RCC_HSE_SetDivision

2.22.11.1 功能介绍

设置 HSE 时钟分频系数。

2.22.11.2 接口定义

函数接口	void LL_RCC_HSE_SetDivision (uint32_t HSEDivision)
输入	uint32_t HSEDivision: { LL_RCC_HSE_DIV_1, LL_RCC_HSE_DIV_2, LL_RCC_HSE_DIV_4, LL_RCC_HSE_DIV_8, LL_RCC_HSE_DIV_16, LL_RCC_HSE_DIV_32 }
输出	无
返回值	无
资源使用	
说明	

2.22.12 LL_RCC_HSE_GetDivision

2.22.12.1 功能介绍

获取 HSE 时钟分频系数。

2.22.12.2 接口定义

函数接口	uint32_t LL_RCC_HSE_GetDivision (void)
输入	无
输出	无
返回值	{ LL_RCC_HSE_DIV_1, LL_RCC_HSE_DIV_2, LL_RCC_HSE_DIV_4, LL_RCC_HSE_DIV_8, LL_RCC_HSE_DIV_16, LL_RCC_HSE_DIV_32 }
资源使用	
说明	

2.22.13 LL_RCC_HSE_SetOutPin

2.22.13.1 功能介绍

设置 HSE XO 输出管脚选择。

2.22.13.2 接口定义

函数接口	void LL_RCC_HSE_SetOutPin (uint32_t OutPin)
输入	uint32_t OutPin: { LL_RCC_HSE_OUTPIN_IO1, LL_RCC_HSE_OUTPIN_IO2 }
输出	无
返回值	无
资源使用	
说明	

2.22.14 LL_RCC_HSE_GetOutPin

2.22.14.1 功能介绍

获取 HSE XO 输出管脚选择。

2.22.14.2 接口定义

函数接口	uint32_t LL_RCC_HSE_GetOutPin (void)
输入	无
输出	无
返回值	{ LL_RCC_HSE_OUTPIN_IO1, LL_RCC_HSE_OUTPIN_IO2 }
资源使用	
说明	

2.22.15 LL_RCC_HSE_SetInPin

2.22.15.1 功能介绍

设置 HSE XI 输入管脚选择。

2.22.15.2 接口定义

函数接口	void LL_RCC_HSE_SetInPin (uint32_t InPin)
------	---

输入	uint32_t InPin: { LL_RCC_HSE_INPIN_IO1, LL_RCC_HSE_INPIN_IO2, LL_RCC_HSE_INPIN_IO3 }
输出	无
返回值	无
资源使用	
说明	

2.22.16 LL_RCC_HSE_GetInPin

2.22.16.1 功能介绍

获取 HSE XI 输入管脚选择。

2.22.16.2 接口定义

函数接口	uint32_t LL_RCC_HSE_GetInPin (void)
输入	无
输出	无
返回值	{ LL_RCC_HSE_INPIN_IO1, LL_RCC_HSE_INPIN_IO2, LL_RCC_HSE_INPIN_IO3 }
资源使用	
说明	

2.22.17 LL_RCC_HSE_SetDrive

2.22.17.1 功能介绍

设置 HSE 驱动能力选择。

2.22.17.2 接口定义

函数接口	void LL_RCC_HSE_SetDrive (uint32_t drive)
输入	uint32_t drive: { LL_RCC_HSE_DRV_NORMAL, LL_RCC_HSE_DRV_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.22.18 LL_RCC_HSE_GetDrive

2.22.18.1 功能介绍

获取 HSE 驱动能力选择。

2.22.18.2 接口定义

函数接口	uint32_t LL_RCC_HSE_GetDrive (void)
输入	无
输出	无
返回值	{ LL_RCC_HSE_DRV_NORMAL, LL_RCC_HSE_DRV_HIGH }
资源使用	

说明	
----	--

2.22.19 LL_RCC_HSE_IsReady

2.22.19.1 功能介绍

检查 HSE 时钟稳定标志。

2.22.19.2 接口定义

函数接口	uint32_t LL_RCC_HSE_IsReady (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.20 LL_RCC_LSE_EnableCSS

2.22.20.1 功能介绍

使能 LSE 时钟安全系统。。

2.22.20.2 接口定义

函数接口	void LL_RCC_LSE_EnableCSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.21 LL_RCC_LSE_DisableCSS

2.22.21.1 功能介绍

禁止使能 LSE 时钟安全系统。。

2.22.21.2 接口定义

函数接口	void LL_RCC_LSE_DisableCSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.22 LL_RCC_LSE_IsEnabledCSS

2.22.22.1 功能介绍

检查是否使能 LSE 时钟安全系统。

2.22.22.2 接口定义

函数接口	uint32_t LL_RCC_LSE_IsEnabledCSS (void)
输入	无

输出	无
返回值	{0,1}
资源使用	
说明	

2.22.23 LL_RCC_LSE_EnableBypass

2.22.23.1 功能介绍

使能 LSE 旁路模式。

2.22.23.2 接口定义

函数接口	void LL_RCC_LSE_EnableBypass (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.24 LL_RCC_LSE_DisableBypass

2.22.24.1 功能介绍

禁止使能 LSE 旁路模式。

2.22.24.2 接口定义

函数接口	void LL_RCC_LSE_DisableBypass (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.25 LL_RCC_LSE_IsEnabledBypass

2.22.25.1 功能介绍

检查是否使能 LSE 旁路模式。

2.22.25.2 接口定义

函数接口	uint32_t LL_RCC_LSE_IsEnabledBypass (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.26 LL_RCC_LSE_Enable

2.22.26.1 功能介绍

使能 LSE。

2.22.26.2 接口定义

函数接口	void LL_RCC_LSE_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.27 LL_RCC_LSE_Disable

2.22.27.1 功能介绍

禁止使能 LSE。

2.22.27.2 接口定义

函数接口	void LL_RCC_LSE_Disable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.28 LL_RCC_LSE_IsEnabled

2.22.28.1 功能介绍

检查是否使能 LSE。

2.22.28.2 接口定义

函数接口	uint32_t LL_RCC_LSE_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.29 LL_RCC_LSE_SetState

2.22.29.1 功能介绍

设置 LSE 状态。

2.22.29.2 接口定义

函数接口	void LL_RCC_LSE_SetState (uint32_t LSEState)
输入	uint32_t LSEState: { LL_RCC_LSE_OFF, LL_RCC_LSE_ON, LL_RCC_LSE_BYPASS }
输出	无
返回值	无
资源使用	
说明	

2.22.30 LL_RCC_LSE_GetState

2.22.30.1 功能介绍

获取 LSE 状态。

2.22.30.2 接口定义

函数接口	uint32_t LL_RCC_LSE_GetState (void)
输入	无
输出	无
返回值	{ LL_RCC_LSE_OFF, LL_RCC_LSE_ON, LL_RCC_LSE_BYPASS }
资源使用	
说明	

2.22.31 LL_RCC_LSE_SetDriveCapability

2.22.31.1 功能介绍

设置 LSE 驱动能力。

2.22.31.2 接口定义

函数接口	void LL_RCC_LSE_SetDriveCapability (uint32_t LSEDrive)
输入	uint32_t LSEDrive: { LL_RCC_LSEDRIVE_LOW, LL_RCC_LSEDRIVE_MEDIUMLOW, LL_RCC_LSEDRIVE_MEDIUMHIGH, LL_RCC_LSEDRIVE_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.22.32 LL_RCC_LSE_GetDriveCapability

2.22.32.1 功能介绍

获取 LSE 驱动能力。

2.22.32.2 接口定义

函数接口	uint32_t LL_RCC_LSE_GetDriveCapability (void)
输入	无
输出	无
返回值	{ LL_RCC_LSEDRIVE_LOW, LL_RCC_LSEDRIVE_MEDIUMLOW, LL_RCC_LSEDRIVE_MEDIUMHIGH, LL_RCC_LSEDRIVE_HIGH }
资源使用	
说明	

2.22.33 LL_RCC_LSE_SetStableCycle

2.22.33.1 功能介绍

设置等待 LSE 时钟稳定的周期数。

2.22.33.2 接口定义

函数接口	void LL_RCC_LSE_SetStableCycle (uint32_t LSEStableCycle)
------	--

输入	uint32_t LSEStableCycle: { LL_RCC_LSE_STABLECYCLE_2, LL_RCC_LSE_STABLECYCLE_1024, LL_RCC_LSE_STABLECYCLE_4096, LL_RCC_LSE_STABLECYCLE_8192, LL_RCC_LSE_STABLECYCLE_16384, LL_RCC_LSE_STABLECYCLE_32768, LL_RCC_LSE_STABLECYCLE_65535, LL_RCC_LSE_STABLECYCLE_131072}
输出	无
返回值	无
资源使用	
说明	

2.22.34 LL_RCC_LSE_GetStableCycle

2.22.34.1 功能介绍

获取等待 LSE 时钟稳定的周期数。

2.22.34.2 接口定义

函数接口	uint32_t LL_RCC_LSE_GetStableCycle (void)
输入	无
输出	无
返回值	{ LL_RCC_LSE_STABLECYCLE_2, LL_RCC_LSE_STABLECYCLE_1024, LL_RCC_LSE_STABLECYCLE_4096, LL_RCC_LSE_STABLECYCLE_8192, LL_RCC_LSE_STABLECYCLE_16384, LL_RCC_LSE_STABLECYCLE_32768, LL_RCC_LSE_STABLECYCLE_65535, LL_RCC_LSE_STABLECYCLE_131072}
资源使用	
说明	

2.22.35 LL_RCC_LSE_IsReady

2.22.35.1 功能介绍

检查 LSE 时钟稳定标志。

2.22.35.2 接口定义

函数接口	uint32_t LL_RCC_LSE_IsReady (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.36 LL_RCC_LSE_IsCSSDetected

2.22.36.1 功能介绍

检查 LSE CSS 故障检测状态指示。

2.22.36.2 接口定义

函数接口	uint32_t LL_RCC_LSE_IsCSSDetected (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.37 LL_RCC_HSI_EnableAlwaysON

2.22.37.1 功能介绍

始终为外设使能 HSI。

2.22.37.2 接口定义

函数接口	void LL_RCC_HSI_AlwaysON (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.38 LL_RCC_HSI_DisableAlwaysON

2.22.38.1 功能介绍

禁止始终为外设使能 HSI。

2.22.38.2 接口定义

函数接口	void LL_RCC_HSI_DisableAlwaysON (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.39 LL_RCC_HSI_IsEnabledAlwaysON

2.22.39.1 功能介绍

检查是否始终为外设使能 HSI。

2.22.39.2 接口定义

函数接口	uint32_t LL_RCC_HSI_IsEnabledAlwaysON (void)
输入	无
输出	无

返回值	{0,1}
资源使用	
说明	

2.22.40 LL_RCC_HSI_Enable

2.22.40.1 功能介绍

使能 HSI。

2.22.40.2 接口定义

函数接口	void LL_RCC_HSI_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.41 LL_RCC_HSI_Disable

2.22.41.1 功能介绍

禁止使能 HSI。

2.22.41.2 接口定义

函数接口	void LL_RCC_HSI_Disable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.42 LL_RCC_HSI_IsEnabled

2.22.42.1 功能介绍

检查是否使能 HSI。

2.22.42.2 接口定义

函数接口	uint32_t LL_RCC_HSI_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.43 LL_RCC_HSI_SetState

2.22.43.1 功能介绍

设置 HSI 状态。

2.22.43.2 接口定义

函数接口	void LL_RCC_HSI_SetState (uint32_t HSISState)
输入	uint32_t HSISState: { LL_RCC_HSI_OFF, LL_RCC_HSI_ON, LL_RCC_HSI_BYPASS }
输出	无
返回值	无
资源使用	
说明	

2.22.44 LL_RCC_HSI_GetState

2.22.44.1 功能介绍

获取 HSI 状态。

2.22.44.2 接口定义

函数接口	uint32_t LL_RCC_HSI_GetState (void)
输入	无
输出	无
返回值	{ LL_RCC_HSI_OFF, LL_RCC_HSI_ON, LL_RCC_HSI_BYPASS }
资源使用	
说明	

2.22.45 LL_RCC_HSI_IsReady

2.22.45.1 功能介绍

检查 HSI 时钟稳定标志。

2.22.45.2 接口定义

函数接口	uint32_t LL_RCC_HSI_IsReady (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.46 LL_RCC_HSI_SetCalibTrimming

2.22.46.1 功能介绍

设置 HSI 微调值。

2.22.46.2 接口定义

函数接口	void LL_RCC_HSI_SetCalibTrimming (uint32_t Value)
输入	uint32_t Value: [0x000, 0x3FF]
输出	无
返回值	无
资源使用	

说明	
----	--

2.22.47 LL_RCC_HSI_GetCalibTrimming

2.22.47.1 功能介绍

获取 HSI 微调值。

2.22.47.2 接口定义

函数接口	uint32_t LL_RCC_HSI_GetCalibTrimming (void)
输入	无
输出	无
返回值	[0x000, 0x3FF]
资源使用	
说明	

2.22.48 LL_RCC_HSI_SetDivision

2.22.48.1 功能介绍

设置 HSI 时钟分频系数。

2.22.48.2 接口定义

函数接口	void LL_RCC_HSI_SetDivision (uint32_t HSIDivision)
输入	uint32_t HSIDivision: {LL_RCC_HSI_DIV_1, LL_RCC_HSI_DIV_2, LL_RCC_HSI_DIV_4, LL_RCC_HSI_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.22.49 LL_RCC_HSI_GetDivision

2.22.49.1 功能介绍

获取 HSI 时钟分频系数。

2.22.49.2 接口定义

函数接口	uint32_t LL_RCC_HSI_GetDivision (void)
输入	无
输出	无
返回值	{LL_RCC_HSI_DIV_1, LL_RCC_HSI_DIV_2, LL_RCC_HSI_DIV_4, LL_RCC_HSI_DIV_8 }
资源使用	
说明	

2.22.50 LL_RCC_HSI_SetStableCycle

2.22.50.1 功能介绍

设置等待 HSI 时钟稳定的周期数。

2.22.50.2 接口定义

函数接口	void LL_RCC_HSI_SetStableCycle (uint32_t HSIStableCycle)
输入	uint32_t HSIStableCycle: { LL_RCC_HSI_STABLECYCLE_2, LL_RCC_HSI_STABLECYCLE_4, LL_RCC_HSI_STABLECYCLE_8, LL_RCC_HSI_STABLECYCLE_16, LL_RCC_HSI_STABLECYCLE_32 }
输出	无
返回值	无
资源使用	
说明	

2.22.51 LL_RCC_HSI_GetStableCycle

2.22.51.1 功能介绍

获取等待 HSI 时钟稳定的周期数。

2.22.51.2 接口定义

函数接口	uint32_t LL_RCC_HSI_GetStableCycle (void)
输入	无
输出	无
返回值	{ LL_RCC_HSI_STABLECYCLE_2, LL_RCC_HSI_STABLECYCLE_4, LL_RCC_HSI_STABLECYCLE_8, LL_RCC_HSI_STABLECYCLE_16, LL_RCC_HSI_STABLECYCLE_32 }
资源使用	
说明	

2.22.52 LL_RCC_LSI_Enable

2.22.52.1 功能介绍

使能 LSI。

2.22.52.2 接口定义

函数接口	void LL_RCC_LSI_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.53 LL_RCC_LSI_Disable

2.22.53.1 功能介绍

禁止使能 LSI。

2.22.53.2 接口定义

函数接口	void LL_RCC_LSI_Disable (void)
输入	无

输出	无
返回值	无
资源使用	
说明	

2.22.54 LL_RCC_LSI_IsEnabled

2.22.54.1 功能介绍

检查是否使能 LSI。

2.22.54.2 接口定义

函数接口	uint32_t LL_RCC_LSI_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.55 LL_RCC_LSI_SetState

2.22.55.1 功能介绍

设置 LSI 状态。

2.22.55.2 接口定义

函数接口	void LL_RCC_LSI_SetState (uint32_t LSISState)
输入	uint32_t LSISState: { LL_RCC_LSI_OFF, LL_RCC_LSI_ON }
输出	无
返回值	无
资源使用	
说明	

2.22.56 LL_RCC_LSI_GetState

2.22.56.1 功能介绍

获取 LSI 状态。

2.22.56.2 接口定义

函数接口	uint32_t LL_RCC_LSI_GetState (void)
输入	无
输出	无
返回值	{ LL_RCC_LSI_OFF, LL_RCC_LSI_ON }
资源使用	
说明	

2.22.57 LL_RCC_LSI_IsReady

2.22.57.1 功能介绍

检查 LSI 时钟稳定标志。

2.22.57.2 接口定义

函数接口	uint32_t LL_RCC_LSI_IsReady (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.58 LL_RCC_LSI_SetCalibTrimming

2.22.58.1 功能介绍

设置 LSI 微调值。

2.22.58.2 接口定义

函数接口	void LL_RCC_LSI_SetCalibTrimming (uint32_t Value)
输入	uint32_t Value: [0, 63]
输出	无
返回值	无
资源使用	
说明	

2.22.59 LL_RCC_LSI_GetCalibTrimming

2.22.59.1 功能介绍

获取 LSI 微调值。

2.22.59.2 接口定义

函数接口	uint32_t LL_RCC_LSI_GetCalibTrimming (void)
输入	无
输出	无
返回值	[0, 63]
资源使用	
说明	

2.22.60 LL_RCC_MSI_Enable

2.22.60.1 功能介绍

使能 MSI。

2.22.60.2 接口定义

函数接口	void LL_RCC_MSI_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.61 LL_RCC_MSI_Disable

2.22.61.1 功能介绍

禁止使能 MSI。

2.22.61.2 接口定义

函数接口	void LL_RCC_MSI_Disable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.62 LL_RCC_MSI_IsEnabled

2.22.62.1 功能介绍

检查是否使能 MSI。

2.22.62.2 接口定义

函数接口	uint32_t LL_RCC_MSI_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.63 LL_RCC_MSI_SetState

2.22.63.1 功能介绍

设置 MSI 状态。

2.22.63.2 接口定义

函数接口	void LL_RCC_MSI_SetState (uint32_t MSISState)
输入	uint32_t MSISState: { LL_RCC_MSI_OFF, LL_RCC_MSI_ON }
输出	无
返回值	无
资源使用	
说明	

2.22.64 LL_RCC_MSI_GetState

2.22.64.1 功能介绍

获取 MSI 状态。

2.22.64.2 接口定义

函数接口	uint32_t LL_RCC_MSI_GetState (void)
输入	无

输出	无
返回值	{ LL_RCC_MSI_OFF, LL_RCC_MSI_ON }
资源使用	
说明	

2.22.65 LL_RCC_MSI_IsReady

2.22.65.1 功能介绍

检查 MSI 时钟稳定标志。

2.22.65.2 接口定义

函数接口	uint32_t LL_RCC_MSI_IsReady (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.66 LL_RCC_MSI_SetCalibTrimming

2.22.66.1 功能介绍

设置 MSI 微调值。

2.22.66.2 接口定义

函数接口	void LL_RCC_MSI_SetCalibTrimming (uint32_t Value)
输入	uint32_t Value: [0, 1023]
输出	无
返回值	无
资源使用	
说明	

2.22.67 LL_RCC_MSI_GetCalibTrimming

2.22.67.1 功能介绍

获取 MSI 微调值。

2.22.67.2 接口定义

函数接口	uint32_t LL_RCC_MSI_GetCalibTrimming (void)
输入	无
输出	无
返回值	[0, 1023]
资源使用	
说明	

2.22.68 LL_RCC_MSI_SetStableCycle

2.22.68.1 功能介绍

设置等待 MSI 时钟稳定的周期数。

2.22.68.2 接口定义

函数接口	void LL_RCC_MSI_SetStableCycle (uint32_t MSIStableCycle)
输入	uint32_t MSIStableCycle: { LL_RCC_MSI_STABLECYCLE_2, LL_RCC_MSI_STABLECYCLE_4, LL_RCC_MSI_STABLECYCLE_8, LL_RCC_MSI_STABLECYCLE_16, LL_RCC_MSI_STABLECYCLE_32 }
输出	无
返回值	无
资源使用	
说明	

2.22.69 LL_RCC_MSI_GetStableCycle

2.22.69.1 功能介绍

获取等待 MSI 时钟稳定的周期数。

2.22.69.2 接口定义

函数接口	uint32_t LL_RCC_MSI_GetStableCycle (void)
输入	无
输出	无
返回值	{ LL_RCC_MSI_STABLECYCLE_2, LL_RCC_MSI_STABLECYCLE_4, LL_RCC_MSI_STABLECYCLE_8, LL_RCC_MSI_STABLECYCLE_16, LL_RCC_MSI_STABLECYCLE_32 }
资源使用	
说明	

2.22.70 LL_RCC_LSCO_Enable

2.22.70.1 功能介绍

使能 LSCO。

2.22.70.2 接口定义

函数接口	void LL_RCC_LSCO_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.71 LL_RCC_LSCO_Disable

2.22.71.1 功能介绍

禁止使能 LSCO。

2.22.71.2 接口定义

函数接口	void LL_RCC_LSCO_Disable (void)
输入	无

输出	无
返回值	无
资源使用	
说明	

2.22.72 LL_RCC_LSCO_IsEnabled

2.22.72.1 功能介绍

检查是否使能 LSCO。

2.22.72.2 接口定义

函数接口	uint32_t LL_RCC_LSCO_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.73 LL_RCC_LSCO_SetSource

2.22.73.1 功能介绍

配置低速时钟源。

2.22.73.2 接口定义

函数接口	void LL_RCC_LSCO_SetSource (uint32_t Source)
输入	uint32_t Source: { LL_RCC_LSCO_CLKSOURCE_LSI, LL_RCC_LSCO_CLKSOURCE_LSE }
输出	无
返回值	无
资源使用	
说明	

2.22.74 LL_RCC_LSCO_GetSource

2.22.74.1 功能介绍

获取低速时钟源。

2.22.74.2 接口定义

函数接口	uint32_t LL_RCC_LSCO_GetSource (void)
输入	无
输出	无
返回值	{ LL_RCC_LSCO_CLKSOURCE_LSI, LL_RCC_LSCO_CLKSOURCE_LSE }
资源使用	
说明	

2.22.75 LL_RCC_PLL_EnableCSS

2.22.75.1 功能介绍

使能 PLL 时钟安全系统。。

2.22.75.2 接口定义

函数接口	void LL_RCC_PLL_EnableCSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.76 LL_RCC_PLL_IsEnabledCSS

2.22.76.1 功能介绍

检查是否使能 PLL 时钟安全系统。。

2.22.76.2 接口定义

函数接口	uint32_t LL_RCC_PLL_IsEnabledCSS (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.77 LL_RCC_PLL_Enable

2.22.77.1 功能介绍

使能 PLL。

2.22.77.2 接口定义

函数接口	void LL_RCC_PLL_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.78 LL_RCC_PLL_Disable

2.22.78.1 功能介绍

禁止使能 PLL。

2.22.78.2 接口定义

函数接口	void LL_RCC_PLL_Disable (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.22.79 LL_RCC_PLL_IsEnabled

2.22.79.1 功能介绍

检查是否使能 PLL。

2.22.79.2 接口定义

函数接口	uint32_t LL_RCC_PLL_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.80 LL_RCC_PLL_IsReady

2.22.80.1 功能介绍

检查 PLL 时钟稳定标志。

2.22.80.2 接口定义

函数接口	uint32_t LL_RCC_PLL_IsReady (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.81 LL_RCC_PLL_EnableDomain_SYS

2.22.81.1 功能介绍

使能 PLL 输出映射到 SYSCLK 域。

2.22.81.2 接口定义

函数接口	void LL_RCC_PLL_EnableDomain_SYS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.82 LL_RCC_PLL_DisableDomain_SYS

2.22.82.1 功能介绍

禁止使能 PLL 启用锁相环输出映射到 SYSCLK 域。

2.22.82.2 接口定义

函数接口	void LL_RCC_PLL_DisableDomain_SYS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.83 LL_RCC_PLL_IsEnabledDomain_SYS

2.22.83.1 功能介绍

检查是否使能 PLL 启用锁相环输出映射到 SYSCLK 域。

2.22.83.2 接口定义

函数接口	uint32_t LL_RCC_PLL_IsEnabledDomain_SYS (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.84 LL_RCC_PLL_EnableDomain_TIM1

2.22.84.1 功能介绍

使能 PLL 输出映射到 TIM1 时钟域。

2.22.84.2 接口定义

函数接口	void LL_RCC_PLL_EnableDomain_TIM1 (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.85 LL_RCC_PLL_DisableDomain_TIM1

2.22.85.1 功能介绍

禁止使能 PLL 输出映射到 TIM1 时钟域。

2.22.85.2 接口定义

函数接口	void LL_RCC_PLL_DisableDomain_TIM1 (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.86 LL_RCC_PLL_IsEnabledDomain_TIM1

2.22.86.1 功能介绍

检查是否使能 PLL 输出映射到 TIM1 时钟域。

2.22.86.2 接口定义

函数接口	uint32_t LL_RCC_PLL_IsEnabledDomain_TIM1 (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.87 LL_RCC_PLL_ConfigDomain_SYS

2.22.87.1 功能介绍

配置 SYSCLK 域的 PLL。

2.22.87.2 接口定义

函数接口	void LL_RCC_PLL_ConfigDomain_SYS (uint32_t Source, uint32_t PLLN, uint32_t PLLM, uint32_t PLLR)
输入	uint32_t Source: { LL_RCC_PLLSOURCE_NONE, LL_RCC_PLLSOURCE_HSE_DIV2, LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLSOURCE_HSE } uint32_t PLLN: { LL_RCC_PLLN_DIV_1, LL_RCC_PLLN_DIV_2, LL_RCC_PLLN_DIV_3, LL_RCC_PLLN_DIV_4, LL_RCC_PLLN_DIV_5, LL_RCC_PLLN_DIV_6, LL_RCC_PLLN_DIV_7, LL_RCC_PLLN_DIV_8 } uint32_t PLLM: [1, 32] uint32_t PLLR: { LL_RCC_PLLR_DIV_1, LL_RCC_PLLR_DIV_2, LL_RCC_PLLR_DIV_3, LL_RCC_PLLR_DIV_4, LL_RCC_PLLR_DIV_5, LL_RCC_PLLR_DIV_6, LL_RCC_PLLR_DIV_7, LL_RCC_PLLR_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.22.88 LL_RCC_PLL_ConfigDomain_TIM1

2.22.88.1 功能介绍

配置 TIM1 时钟域的 PLL。

2.22.88.2 接口定义

函数接口	void LL_RCC_PLL_ConfigDomain_TIM1 (uint32_t Source, uint32_t
------	---

	PLLN, uint32_t PLLM, uint32_t PLLQ)
输入	uint32_t Source: { LL_RCC_PLLSOURCE_NONE, LL_RCC_PLLSOURCE_HSE_DIV2, LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLSOURCE_HSE } int32_t PLLN: { LL_RCC_PLLN_DIV_1, LL_RCC_PLLN_DIV_2, LL_RCC_PLLN_DIV_3, LL_RCC_PLLN_DIV_4, LL_RCC_PLLN_DIV_5, LL_RCC_PLLN_DIV_6, LL_RCC_PLLN_DIV_7, LL_RCC_PLLN_DIV_8 } uint32_t PLLM: [1, 32] uint32_t PLLQ: { LL_RCC_PLLQ_DIV_1, LL_RCC_PLLQ_DIV_2, LL_RCC_PLLQ_DIV_3, LL_RCC_PLLQ_DIV_4, LL_RCC_PLLQ_DIV_5, LL_RCC_PLLQ_DIV_6, LL_RCC_PLLQ_DIV_7, LL_RCC_PLLQ_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.22.89 LL_RCC_PLL_GetM

2.22.89.1 功能介绍

获取 PLL 输入时钟的倍频因子 M。

2.22.89.2 接口定义

函数接口	uint32_t LL_RCC_PLL_GetM (void)
输入	无
输出	无
返回值	[1, 32]
资源使用	
说明	

2.22.90 LL_RCC_PLL_SetM

2.22.90.1 功能介绍

设置 PLL 输入时钟的倍频因子 M。

2.22.90.2 接口定义

函数接口	void LL_RCC_PLL_SetM (uint32_t PLLM)
输入	uint32_t PLLM: [1, 32]
输出	无
返回值	无
资源使用	

说明	
----	--

2.22.91 LL_RCC_PLL_GetR

2.22.91.1 功能介绍

获取 PLL VCO 的 R 分频系数。

2.22.91.2 接口定义

函数接口	uint32_t LL_RCC_PLL_GetR (void)
输入	无
输出	无
返回值	{ LL_RCC_PLLR_DIV_1, LL_RCC_PLLR_DIV_2, LL_RCC_PLLR_DIV_3, LL_RCC_PLLR_DIV_4, LL_RCC_PLLR_DIV_5, LL_RCC_PLLR_DIV_6, LL_RCC_PLLR_DIV_7, LL_RCC_PLLR_DIV_8 }
资源使用	
说明	

2.22.92 LL_RCC_PLL_SetR

2.22.92.1 功能介绍

设置 PLL VCO 的 R 分频系数。

2.22.92.2 接口定义

函数接口	void LL_RCC_PLL_SetR (uint32_t PLLRDivision)
输入	uint32_t PLLRDivision: { LL_RCC_PLLR_DIV_1, LL_RCC_PLLR_DIV_2, LL_RCC_PLLR_DIV_3, LL_RCC_PLLR_DIV_4, LL_RCC_PLLR_DIV_5, LL_RCC_PLLR_DIV_6, LL_RCC_PLLR_DIV_7, LL_RCC_PLLR_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.22.93 LL_RCC_PLL_GetQ

2.22.93.1 功能介绍

获取 PLL VCO 的 Q 分频系数。

2.22.93.2 接口定义

函数接口	uint32_t LL_RCC_PLL_GetQ (void)
输入	无
输出	无
返回值	{ LL_RCC_PLLQ_DIV_1, LL_RCC_PLLQ_DIV_2, LL_RCC_PLLQ_DIV_3, LL_RCC_PLLQ_DIV_4, LL_RCC_PLLQ_DIV_5, LL_RCC_PLLQ_DIV_6, LL_RCC_PLLQ_DIV_7, LL_RCC_PLLQ_DIV_8 }

资源使用	
说明	

2.22.94 LL_RCC_PLL_SetQ

2.22.94.1 功能介绍

设置 PLL VCO 的 Q 分频系数。

2.22.94.2 接口定义

函数接口	void LL_RCC_PLL_SetQ (uint32_t PLLQDivision)
输入	uint32_t PLLQDivision: { LL_RCC_PLLQ_DIV_1, LL_RCC_PLLQ_DIV_2, LL_RCC_PLLQ_DIV_3, LL_RCC_PLLQ_DIV_4, LL_RCC_PLLQ_DIV_5, LL_RCC_PLLQ_DIV_6, LL_RCC_PLLQ_DIV_7, LL_RCC_PLLQ_DIV_8 }
输出	无
返回值	无
资源使用	
说明	

2.22.95 LL_RCC_PLL_GetN

2.22.95.1 功能介绍

获取 PLL 输入时钟的分频因子 N。

2.22.95.2 接口定义

函数接口	uint32_t LL_RCC_PLL_GetN (void)
输入	无
输出	无
返回值	{ LL_RCC_PLLN_DIV_1, LL_RCC_PLLN_DIV_2, LL_RCC_PLLN_DIV_3, LL_RCC_PLLN_DIV_4, LL_RCC_PLLN_DIV_5, LL_RCC_PLLN_DIV_6, LL_RCC_PLLN_DIV_7, LL_RCC_PLLN_DIV_8 }
资源使用	
说明	

2.22.96 LL_RCC_PLL_SetN

2.22.96.1 功能介绍

设置 PLL 输入时钟的分频因子 N。

2.22.96.2 接口定义

函数接口	void LL_RCC_PLL_SetN (uint32_t PLLN)
输入	uint32_t PLLN: { LL_RCC_PLLN_DIV_1, LL_RCC_PLLN_DIV_2, LL_RCC_PLLN_DIV_3, LL_RCC_PLLN_DIV_4, LL_RCC_PLLN_DIV_5, LL_RCC_PLLN_DIV_6, LL_RCC_PLLN_DIV_7, LL_RCC_PLLN_DIV_8 }

输出	无
返回值	无
资源使用	
说明	

2.22.97 LL_RCC_PLL_GetMainSource

2.22.97.1 功能介绍

获取 PLL 输入时钟源。

2.22.97.2 接口定义

函数接口	uint32_t LL_RCC_PLL_GetMainSource (void)
输入	无
输出	无
返回值	{ LL_RCC_PLLSOURCE_NONE, LL_RCC_PLLSOURCE_HSE_DIV2, LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLSOURCE_HSE }
资源使用	
说明	

2.22.98 LL_RCC_PLL_SetMainSource

2.22.98.1 功能介绍

设置 PLL 输入时钟源。

2.22.98.2 接口定义

函数接口	void LL_RCC_PLL_SetMainSource (uint32_t PLLSource)
输入	int32_t PLLSource: { LL_RCC_PLLSOURCE_NONE, LL_RCC_PLLSOURCE_HSE_DIV2, LL_RCC_PLLSOURCE_HSI, LL_RCC_PLLSOURCE_HSE }
输出	无
返回值	无
资源使用	
说明	

2.22.99 LL_RCC_SetSysClkSource

2.22.99.1 功能介绍

设置系统时钟源。

2.22.99.2 接口定义

函数接口	void LL_RCC_SetSysClkSource (uint32_t Source)
输入	int32_t Source: { LL_RCC_SYS_CLKSOURCE_HSYSYS, LL_RCC_SYS_CLKSOURCE_HSE, LL_RCC_SYS_CLKSOURCE_PLL, LL_RCC_SYS_CLKSOURCE_LSI, LL_RCC_SYS_CLKSOURCE_LSE, LL_RCC_SYS_CLKSOURCE_MSI }
输出	无
返回值	无

资源使用	
说明	

2.22.100 LL_RCC_GetSysClkSource

2.22.100.1 功能介绍

获取系统时钟源。

2.22.100.2 接口定义

函数接口	uint32_t LL_RCC_GetSysClkSource (void)
输入	无
输出	无
返回值	{ LL_RCC_SYS_CLKSOURCE_HSYSYS, LL_RCC_SYS_CLKSOURCE_HSE, LL_RCC_SYS_CLKSOURCE_PLL, LL_RCC_SYS_CLKSOURCE_LSI, LL_RCC_SYS_CLKSOURCE_LSE, LL_RCC_SYS_CLKSOURCE_MSI }
资源使用	
说明	

2.22.101 LL_RCC_SetAHBPrescaler

2.22.101.1 功能介绍

设置 AHB 预分频器。

2.22.101.2 接口定义

函数接口	void LL_RCC_SetAHBPrescaler (uint32_t Prescaler)
输入	uint32_t Prescaler: { LL_RCC_SYSCLK_DIV_1, LL_RCC_SYSCLK_DIV_2, LL_RCC_SYSCLK_DIV_4, LL_RCC_SYSCLK_DIV_8, LL_RCC_SYSCLK_DIV_16, LL_RCC_SYSCLK_DIV_32, LL_RCC_SYSCLK_DIV_64, LL_RCC_SYSCLK_DIV_128 }
输出	无
返回值	无
资源使用	
说明	

2.22.102 LL_RCC_GetAHBPrescaler

2.22.102.1 功能介绍

获取 AHB 预分频器。

2.22.102.2 接口定义

函数接口	uint32_t LL_RCC_GetAHBPrescaler (void)
输入	无
输出	无
返回值	{ LL_RCC_SYSCLK_DIV_1, LL_RCC_SYSCLK_DIV_2, LL_RCC_SYSCLK_DIV_4, LL_RCC_SYSCLK_DIV_8, LL_RCC_SYSCLK_DIV_16, LL_RCC_SYSCLK_DIV_32,

	LL_RCC_SYSCLK_DIV_64, LL_RCC_SYSCLK_DIV_128 }
资源使用	
说明	

2.22.103 LL_RCC_SetAPB1Prescaler

2.22.103.1 功能介绍

设置 APB1 预分频器。

2.22.103.2 接口定义

函数接口	void LL_RCC_SetAPB1Prescaler (uint32_t Prescaler)
输入	int32_t Prescaler: { LL_RCC_APB1_DIV_1, LL_RCC_APB1_DIV_2, LL_RCC_APB1_DIV_4, LL_RCC_APB1_DIV_8, LL_RCC_APB1_DIV_16 }
输出	无
返回值	无
资源使用	
说明	

2.22.104 LL_RCC_GetAPB1Prescaler

2.22.104.1 功能介绍

获取 APB1 预分频器。

2.22.104.2 接口定义

函数接口	uint32_t LL_RCC_GetAPB1Prescaler (void)
输入	无
输出	无
返回值	{ LL_RCC_APB1_DIV_1, LL_RCC_APB1_DIV_2, LL_RCC_APB1_DIV_4, LL_RCC_APB1_DIV_8, LL_RCC_APB1_DIV_16 }
资源使用	
说明	

2.22.105 LL_RCC_SetAPB2Prescaler

2.22.105.1 功能介绍

设置 APB2 预分频器。

2.22.105.2 接口定义

函数接口	void LL_RCC_SetAPB2Prescaler (uint32_t Prescaler)
输入	int32_t Prescaler: { LL_RCC_APB2_DIV_1, LL_RCC_APB2_DIV_2, LL_RCC_APB2_DIV_4, LL_RCC_APB2_DIV_8, LL_RCC_APB2_DIV_16 }
输出	无
返回值	无

资源使用	
说明	

2.22.106 LL_RCC_GetAPB2Prescaler

2.22.106.1 功能介绍

获取 APB2 预分频器。

2.22.106.2 接口定义

函数接口	uint32_t LL_RCC_GetAPB2Prescaler (void)
输入	无
输出	无
返回值	{ LL_RCC_APB2_DIV_1, LL_RCC_APB2_DIV_2, LL_RCC_APB2_DIV_4, LL_RCC_APB2_DIV_8, LL_RCC_APB2_DIV_16 }
资源使用	
说明	

2.22.107 LL_RCC_ConfigMCO

2.22.107.1 功能介绍

配置 MCOx。

2.22.107.2 接口定义

函数接口	void LL_RCC_ConfigMCO (uint32_t MCOxSource, uint32_t MCOxPrescaler)
输入	uint32_t MCOxSource: { LL_RCC_MCOSOURCE_NOCLOCK, LL_RCC_MCOSOURCE_SYS, LL_RCC_MCOSOURCE_MSI, LL_RCC_MCOSOURCE_HSI, LL_RCC_MCOSOURCE_HSE, LL_RCC_MCOSOURCE_PLL, LL_RCC_MCOSOURCE_LSI, LL_RCC_MCOSOURCE_LSE } uint32_t MCOxPrescaler: { LL_RCC_MCO_DIV_1, LL_RCC_MCO_DIV_2, LL_RCC_MCO_DIV_4, LL_RCC_MCO_DIV_8, LL_RCC_MCO_DIV_16, LL_RCC_MCO_DIV_32, LL_RCC_MCO_DIV_64, LL_RCC_MCO_DIV_128 }
输出	无
返回值	无
资源使用	
说明	

2.22.108 LL_RCC_GetMCOPrescaler

2.22.108.1 功能介绍

获取 MCOx 预分频器。

2.22.108.2 接口定义

函数接口	uint32_t LL_RCC_GetMCOPrescaler (void)
输入	无

输出	无
返回值	{LL_RCC_MCO_DIV_1, LL_RCC_MCO_DIV_2, LL_RCC_MCO_DIV_4, LL_RCC_MCO_DIV_8, LL_RCC_MCO_DIV_16, LL_RCC_MCO_DIV_32, LL_RCC_MCO_DIV_64, LL_RCC_MCO_DIV_128 }
资源使用	
说明	

2.22.109 LL_RCC_GetMCOSource

2.22.109.1 功能介绍

获取 MCOx 时钟源。

2.22.109.2 接口定义

函数接口	uint32_t LL_RCC_GetMCOSource (void)
输入	无
输出	无
返回值	{ LL_RCC_MCOSOURCE_NOCLOCK, LL_RCC_MCOSOURCE_SYS, LL_RCC_MCOSOURCE_MSI, LL_RCC_MCOSOURCE_HSI, LL_RCC_MCOSOURCE_HSE, LL_RCC_MCOSOURCE_PLL, LL_RCC_MCOSOURCE_LSI, LL_RCC_MCOSOURCE_LSE }
资源使用	
说明	

2.22.110 LL_RCC_SetUSARTClockSource

2.22.110.1 功能介绍

设置 USARTx 时钟源。

2.22.110.2 接口定义

函数接口	void LL_RCC_SetUSARTClockSource (uint32_t USARTx, uint32_t USARTxSource)
输入	uint32_t USARTx: {LL_RCC_USART1_CLKSOURCE} uint32_t USARTxSource: { LL_RCC_USART1_CLKSOURCE_PCLK2, LL_RCC_USART1_CLKSOURCE_SYS, LL_RCC_USART1_CLKSOURCE_HSI, LL_RCC_USART1_CLKSOURCE_LSE }
输出	无
返回值	无
资源使用	
说明	

2.22.111 LL_RCC_GetUSARTClockSource

2.22.111.1 功能介绍

获取 USARTx 时钟源。

2.22.111.2 接口定义

函数接口	uint32_t LL_RCC_GetUSARTClockSource (uint32_t USARTx)
输入	uint32_t USARTx: { LL_RCC_USART1_CLKSOURCE }
输出	无
返回值	{ LL_RCC_USART1_CLKSOURCE_PCLK2, LL_RCC_USART1_CLKSOURCE_SYS, LL_RCC_USART1_CLKSOURCE_HSI, LL_RCC_USART1_CLKSOURCE_LSE }
资源使用	
说明	

2.22.112 LL_RCC_SetLPUARTClockSource

2.22.112.1 功能介绍

设置 LPUARTx 时钟源。

2.22.112.2 接口定义

函数接口	void LL_RCC_SetLPUARTClockSource (uint32_t LPUARTx, uint32_t LPUARTxSource)
输入	uint32_t LPUARTx: { LL_RCC_LPUART1_CLKSOURCE } uint32_t LPUARTxSource: { LL_RCC_LPUART1_CLKSOURCE_PCLK1, LL_RCC_LPUART1_CLKSOURCE_SYS, LL_RCC_LPUART1_CLKSOURCE_HSI, LL_RCC_LPUART1_CLKSOURCE_LSE }
输出	无
返回值	无
资源使用	
说明	

2.22.113 LL_RCC_GetLPUARTClockSource

2.22.113.1 功能介绍

获取 LPUARTx 时钟源。

2.22.113.2 接口定义

函数接口	uint32_t LL_RCC_GetLPUARTClockSource (uint32_t LPUARTx)
输入	uint32_t LPUARTx: { LL_RCC_LPUART1_CLKSOURCE }
输出	无
返回值	{ LL_RCC_LPUART1_CLKSOURCE_PCLK1, LL_RCC_LPUART1_CLKSOURCE_SYS, LL_RCC_LPUART1_CLKSOURCE_HSI, LL_RCC_LPUART1_CLKSOURCE_LSE }
资源使用	

说明	
----	--

2.22.114 LL_RCC_SetI2CClockSource

2.22.114.1 功能介绍

设置 I2Cx 时钟源。

2.22.114.2 接口定义

函数接口	void LL_RCC_SetI2CClockSource (uint32_t I2Cx, uint32_t I2CSource)
输入	uint32_t I2Cx: { LL_RCC_LPUART1_CLKSOURCE } uint32_t I2CSource: { LL_RCC_I2C1_CLKSOURCE_PCLK1, LL_RCC_I2C1_CLKSOURCE_SYS, LL_RCC_I2C1_CLKSOURCE_HSI }
输出	无
返回值	无
资源使用	
说明	

2.22.115 LL_RCC_GetI2CClockSource

2.22.115.1 功能介绍

获取 I2Cx 时钟源。

2.22.115.2 接口定义

函数接口	uint32_t LL_RCC_GetI2CClockSource (uint32_t I2Cx)
输入	uint32_t I2Cx: { LL_RCC_I2C1_CLKSOURCE }
输出	无
返回值	{ LL_RCC_I2C1_CLKSOURCE_PCLK1, LL_RCC_I2C1_CLKSOURCE_SYS, LL_RCC_I2C1_CLKSOURCE_HSI }
资源使用	
说明	

2.22.116 LL_RCC_SetLPTIMClockSource

2.22.116.1 功能介绍

设置 LPTIMx 时钟源。

2.22.116.2 接口定义

函数接口	void LL_RCC_SetLPTIMClockSource (uint32_t LPTIMx, uint32_t LPTIMxSource)
输入	uint32_t LPTIMx: { LL_RCC_LPTIM1_CLKSOURCE } uint32_t LPTIMxSource: { LL_RCC_LPTIM1_CLKSOURCE_PCLK1, LL_RCC_LPTIM1_CLKSOURCE_LSI,

	LL_RCC_LPTIM1_CLKSOURCE_HSI, LL_RCC_LPTIM1_CLKSOURCE_LSE, LL_RCC_LPTIM1_CLKSOURCE_OSC1K }
输出	无
返回值	无
资源使用	
说明	

2.22.117 LL_RCC_GetLPTIMClockSource

2.22.117.1 功能介绍

获取 LPTIMx 时钟源。

2.22.117.2 接口定义

函数接口	uint32_t LL_RCC_GetLPTIMClockSource (uint32_t LPTIMx)
输入	uint32_t LPTIMx: { LL_RCC_LPTIM1_CLKSOURCE }
输出	无
返回值	{ LL_RCC_LPTIM1_CLKSOURCE_PCLK1, LL_RCC_LPTIM1_CLKSOURCE_LSI, LL_RCC_LPTIM1_CLKSOURCE_HSI, LL_RCC_LPTIM1_CLKSOURCE_LSE, LL_RCC_LPTIM1_CLKSOURCE_OSC1K }
资源使用	
说明	

2.22.118 LL_RCC_SetTIMClockSource

2.22.118.1 功能介绍

设置 TIMx 时钟源。

2.22.118.2 接口定义

函数接口	void LL_RCC_SetTIMClockSource (uint32_t TIMx, uint32_t TIMxSource)
输入	uint32_t TIMx: { LL_RCC_TIM1_CLKSOURCE } uint32_t TIMxSource: { LL_RCC_TIM1_CLKSOURCE_PLL, LL_RCC_TIM1_CLKSOURCE_PCLK2 }
输出	无
返回值	无
资源使用	
说明	

2.22.119 LL_RCC_GetTIMClockSource

2.22.119.1 功能介绍

获取 TIMx 时钟源。

2.22.119.2 接口定义

函数接口	uint32_t LL_RCC_GetTIMClockSource (uint32_t TIMx)
输入	uint32_t TIMx: { LL_RCC_TIM1_CLKSOURCE }
输出	无
返回值	{ LL_RCC_TIM1_CLKSOURCE_PLL, LL_RCC_TIM1_CLKSOURCE_PCLK2 }
资源使用	
说明	

2.22.120 LL_RCC_SetADCClockSource

2.22.120.1 功能介绍

设置 ADCx 时钟源。

2.22.120.2 接口定义

函数接口	void LL_RCC_SetADCClockSource (uint32_t ADCx, uint32_t ADCxSource)
输入	uint32_t ADCx: { LL_RCC_ADC1_CLKSOURCE } uint32_t ADCxSource: { LL_RCC_ADC1_CLKSOURCE_SYS, LL_RCC_ADC1_CLKSOURCE_SYS_DIV2, LL_RCC_ADC1_CLKSOURCE_SYS_DIV4, LL_RCC_ADC1_CLKSOURCE_HSI }
输出	无
返回值	无
资源使用	
说明	

2.22.121 LL_RCC_GetADCClockSource

2.22.121.1 功能介绍

获取 ADCx 时钟源。

2.22.121.2 接口定义

函数接口	uint32_t LL_RCC_GetADCClockSource (uint32_t ADCx)
输入	uint32_t ADCx: { LL_RCC_ADC1_CLKSOURCE }
输出	无
返回值	{ LL_RCC_ADC1_CLKSOURCE_SYS, LL_RCC_ADC1_CLKSOURCE_SYS_DIV2, LL_RCC_ADC1_CLKSOURCE_SYS_DIV4, LL_RCC_ADC1_CLKSOURCE_HSI }
资源使用	
说明	

2.22.122 LL_RCC_RTC_SetClockSource

2.22.122.1 功能介绍

设置 RTC 时钟源。

2.22.122.2 接口定义

函数接口	void LL_RCC_RTC_SetClockSource (uint32_t Source)
输入	uint32_t Source: { LL_RCC_RTC_CLKSOURCE_NONE, LL_RCC_RTC_CLKSOURCE_LSE, LL_RCC_RTC_CLKSOURCE_LSI, LL_RCC_RTC_CLKSOURCE_HSE_DIV128, LL_RCC_RTC_CLKSOURCE_HSE_DIV256, LL_RCC_RTC_CLKSOURCE_HSE_DIV512, LL_RCC_RTC_CLKSOURCE_HSE_DIV1024 }
输出	无
返回值	无
资源使用	
说明	

2.22.123 LL_RCC_RTC_GetClockSource

2.22.123.1 功能介绍

获取 RTC 时钟源。

2.22.123.2 接口定义

函数接口	uint32_t LL_RCC_RTC_GetClockSource (void)
输入	无
输出	无
返回值	{ LL_RCC_RTC_CLKSOURCE_NONE, LL_RCC_RTC_CLKSOURCE_LSE, LL_RCC_RTC_CLKSOURCE_LSI, LL_RCC_RTC_CLKSOURCE_HSE_DIV128, LL_RCC_RTC_CLKSOURCE_HSE_DIV256, LL_RCC_RTC_CLKSOURCE_HSE_DIV512, LL_RCC_RTC_CLKSOURCE_HSE_DIV1024 }
资源使用	
说明	

2.22.124 LL_RCC_RTC_SetFreqAdjust

2.22.124.1 功能介绍

设置 RTC 高速时钟补偿频率。

2.22.124.2 接口定义

函数接口	void LL_RCC_RTC_SetFreqAdjust (uint32_t Frequency)
输入	uint32_t Frequency: { LL_RCC_RTC_FREQADJUST_4M, LL_RCC_RTC_FREQADJUST_6M, LL_RCC_RTC_FREQADJUST_8M, LL_RCC_RTC_FREQADJUST_12M,

	LL_RCC_RTC_FREQADJUST_16M, LL_RCC_RTC_FREQADJUST_20M, LL_RCC_RTC_FREQADJUST_24M, LL_RCC_RTC_FREQADJUST_32M }
输出	无
返回值	无
资源使用	
说明	

2.22.125 LL_RCC_RTC_GetFreqAdjust

2.22.125.1 功能介绍

获取 RTC 高速时钟补偿频率。

2.22.125.2 接口定义

函数接口	uint32_t LL_RCC_RTC_GetFreqAdjust (void)
输入	无
输出	无
返回值	{ LL_RCC_RTC_FREQADJUST_4M, LL_RCC_RTC_FREQADJUST_6M, LL_RCC_RTC_FREQADJUST_8M, LL_RCC_RTC_FREQADJUST_12M, LL_RCC_RTC_FREQADJUST_16M, LL_RCC_RTC_FREQADJUST_20M, LL_RCC_RTC_FREQADJUST_24M, LL_RCC_RTC_FREQADJUST_32M }
资源使用	
说明	

2.22.126 LL_RCC_RTC_Enable

2.22.126.1 功能介绍

使能 RTC。

2.22.126.2 接口定义

函数接口	void LL_RCC_RTC_Enable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.127 LL_RCC_RTC_Disable

2.22.127.1 功能介绍

禁止使能 RTC。

2.22.127.2 接口定义

函数接口	void LL_RCC_RTC_Disable (void)
------	----------------------------------

输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.128 LL_RCC_RTC_IsEnabled

2.22.128.1 功能介绍

检查是否使能 RTC。

2.22.128.2 接口定义

函数接口	uint32_t LL_RCC_RTC_IsEnabled (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.129 LL_RCC_RTC_ForceReset

2.22.129.1 功能介绍

强制 RTC 复位。

2.22.129.2 接口定义

函数接口	void LL_RCC_RTC_ForceReset (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.130 LL_RCC_RTC_ReleaseReset

2.22.130.1 功能介绍

释放 RTC 复位。

2.22.130.2 接口定义

函数接口	void LL_RCC_RTC_ReleaseReset (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.131 LL_RCC_RTC_EnableLowpowerMode

2.22.131.1 功能介绍

使能 RTC 低功耗模式。

2.22.131.2 接口定义

函数接口	void LL_RCC_RTC_EnableLowpowerMode (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.132 LL_RCC_RTC_DisableLowpowerMode

2.22.132.1 功能介绍

禁止使能 RTC 低功耗模式。

2.22.132.2 接口定义

函数接口	void LL_RCC_RTC_DisableLowpowerMode (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.133 LL_RCC_RTC_IsEnabledLowpowerMode

2.22.133.1 功能介绍

检查是否使能 RTC 低功耗模式。

2.22.133.2 接口定义

函数接口	uint32_t LL_RCC_RTC_IsEnabledLowpowerMode (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.134 LL_RCC_IsActiveRSTFlag_WWDG

2.22.134.1 功能介绍

检查窗口看门狗复位标志。

2.22.134.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_WWDG (void)
输入	无
输出	无

返回值	{0,1}
资源使用	
说明	

2.22.135 LL_RCC_IsActiveRSTFlag_IWDG

2.22.135.1 功能介绍

检查独立看门狗复位标志。

2.22.135.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_IWDG (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.136 LL_RCC_IsActiveRSTFlag_SFT

2.22.136.1 功能介绍

检查软件复位标志。

2.22.136.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_SFT (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.137 LL_RCC_IsActiveRSTFlag_SFT

2.22.137.1 功能介绍

检查软件复位标志。

2.22.137.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_SFT (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.138 LL_RCC_IsActiveRSTFlag_POR

2.22.138.1 功能介绍

检查 POR/PDR 复位标志。

2.22.138.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_POR (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.139 LL_RCC_IsActiveRSTFlag_NRST

2.22.139.1 功能介绍

检查 NRST 引脚复位标志。

2.22.139.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_NRST (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.140 LL_RCC_IsActiveRSTFlag_OBL

2.22.140.1 功能介绍

检查选项字节加载复位标志。

2.22.140.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_OBL (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.141 LL_RCC_IsActiveRSTFlag_LOCKUP

2.22.141.1 功能介绍

检查 LOCKUP 复位标志。

2.22.141.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_LOCKUP (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.142 LL_RCC_IsActiveRSTFlag_BOR

2.22.142.1 功能介绍

检查 BOR 复位标志。

2.22.142.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag_BOR (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.143 LL_RCC_IsActiveRSTFlag

2.22.143.1 功能介绍

检查复位标志位。

2.22.143.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveRSTFlag (uint32_t Flag)
输入	uint32_t Flag: { LL_RCC_RSTFLAG_WWDG, LL_RCC_RSTFLAG_IWDG, LL_RCC_RSTFLAG_SFT, LL_RCC_RSTFLAG_PWR, LL_RCC_RSTFLAG_NRST, LL_RCC_RSTFLAG_OBL, LL_RCC_RSTFLAG_LOCKUP, LL_RCC_RSTFLAG_BOR }
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.144 LL_RCC_ClearRSTFlag

2.22.144.1 功能介绍

清除所有复位标志位。

2.22.144.2 接口定义

函数接口	void LL_RCC_ClearRSTFlag (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.145 LL_RCC_EnableLOCKUPRST

2.22.145.1 功能介绍

使能 LOCKUP 复位。

2.22.145.2 接口定义

函数接口	void LL_RCC_EnableLOCKUPRST (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.146 LL_RCC_DisableLOCKUPRST

2.22.146.1 功能介绍

禁止 LOCKUP 复位。

2.22.146.2 接口定义

函数接口	void LL_RCC_DisableLOCKUPRST (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.147 LL_RCC_IsEnabledLOCKUPRST

2.22.147.1 功能介绍

检查是否 CKUP 复位。

2.22.147.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledLOCKUPRST (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.148 LL_RCC_ClearFlag_LSIRDY

2.22.148.1 功能介绍

清除 LSI 时钟稳定中断标志

2.22.148.2 接口定义

函数接口	void LL_RCC_ClearFlag_LSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.149 LL_RCC_ClearFlag_LSERDY

2.22.149.1 功能介绍

清除 LSE 钟稳定中断标志

2.22.149.2 接口定义

函数接口	void LL_RCC_ClearFlag_LSERDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.150 LL_RCC_ClearFlag_MSIRDY

2.22.150.1 功能介绍

清除 MSI 时钟稳定中断标志

2.22.150.2 接口定义

函数接口	void LL_RCC_ClearFlag_MSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.151 LL_RCC_ClearFlag_HSIRDY

2.22.151.1 功能介绍

清除 HSI 时钟稳定中断标志

2.22.151.2 接口定义

函数接口	void LL_RCC_ClearFlag_HSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.152 LL_RCC_ClearFlag_HSERDY

2.22.152.1 功能介绍

清除 HSE 时钟稳定中断标志

2.22.152.2 接口定义

函数接口	void LL_RCC_ClearFlag_HSERDY (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.22.153 LL_RCC_ClearFlag_PLLRDY

2.22.153.1 功能介绍

清除 PLL 时钟稳定中断标志

2.22.153.2 接口定义

函数接口	void LL_RCC_ClearFlag_PLLRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.154 LL_RCC_ClearFlag_LSECSS

2.22.154.1 功能介绍

清除 LSE CSS 中断标志

2.22.154.2 接口定义

函数接口	void LL_RCC_ClearFlag_LSECSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.155 LL_RCC_ClearFlag_PLLCSS

2.22.155.1 功能介绍

清除 PLL CSS 中断标志

2.22.155.2 接口定义

函数接口	void LL_RCC_ClearFlag_PLLCSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.156 LL_RCC_ClearFlag

2.22.156.1 功能介绍

清除中断标志

2.22.156.2 接口定义

函数接口	void LL_RCC_ClearFlag (uint32_t Flag)
输入	uint32_t Flag: { LL_RCC_FLAG_PLLCSS, LL_RCC_FLAG_LSECSS, LL_RCC_FLAG_PLLRDY, LL_RCC_FLAG_HSERDY, LL_RCC_FLAG_HSIRDY, LL_RCC_FLAG_MSIRDY, LL_RCC_FLAG_LSERDY, LL_RCC_FLAG_LSIRDY }
输出	无
返回值	无
资源使用	
说明	

2.22.157 LL_RCC_IsActiveFlag_LSIRDY

2.22.157.1 功能介绍

检查 LSI 时钟稳定中断标志。

2.22.157.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_LSIRDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.158 LL_RCC_IsActiveFlag_LSERDY

2.22.158.1 功能介绍

检查 LSE 时钟稳定中断标志。

2.22.158.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_LSERDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.159 LL_RCC_IsActiveFlag_MSIRDY

2.22.159.1 功能介绍

检查 MSI 时钟稳定中断标志。

2.22.159.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_MSIRDY (void)
输入	无
输出	无

返回值	{0,1}
资源使用	
说明	

2.22.160 LL_RCC_IsActiveFlag_HSIRDY

2.22.160.1 功能介绍

检查 HSI 时钟稳定中断标志。

2.22.160.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_HSIRDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.161 LL_RCC_IsActiveFlag_HSERDY

2.22.161.1 功能介绍

检查 HSE 时钟稳定中断标志。

2.22.161.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_HSERDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.162 LL_RCC_IsActiveFlag_PLLRDY

2.22.162.1 功能介绍

检查 PLL 时钟稳定中断标志。

2.22.162.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_PLLRDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.163 LL_RCC_IsActiveFlag_LSECSS

2.22.163.1 功能介绍

检查 LSE CSS 中断标志。

2.22.163.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_LSECSS (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.164 LL_RCC_IsActiveFlag_PLLCSS

2.22.164.1 功能介绍

检查 PLL CSS 中断标志。

2.22.164.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag_PLLCSS (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.165 LL_RCC_IsActiveFlag

2.22.165.1 功能介绍

检查中断标志。

2.22.165.2 接口定义

函数接口	uint32_t LL_RCC_IsActiveFlag (uint32_t Flag)
输入	uint32_t Flag: { LL_RCC_FLAG_PLLCSS, LL_RCC_FLAG_LSECSS, LL_RCC_FLAG_PLLRDY, LL_RCC_FLAG_HSERDY, LL_RCC_FLAG_HSIRDY, LL_RCC_FLAG_MSIRDY, LL_RCC_FLAG_LSERDY, LL_RCC_FLAG_LSIRDY }
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.166 LL_RCC_EnableIT_LSIRDY

2.22.166.1 功能介绍

使能 LSI 稳定中断。

2.22.166.2 接口定义

函数接口	void LL_RCC_EnableIT_LSIRDY (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

2.22.167 LL_RCC_DisableIT_LSIRDY

2.22.167.1 功能介绍

禁止使能 LSI 稳定中断。

2.22.167.2 接口定义

函数接口	void LL_RCC_DisableIT_LSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.168 LL_RCC_IsEnabledIT_LSIRDY

2.22.168.1 功能介绍

检查是否使能 LSI 稳定中断。

2.22.168.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT_LSIRDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.169 LL_RCC_EnableIT_LSERDY

2.22.169.1 功能介绍

使能 LSE 稳定中断。

2.22.169.2 接口定义

函数接口	void LL_RCC_EnableIT_LSERDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.170 LL_RCC_DisableIT_LSERDY

2.22.170.1 功能介绍

禁止使能 LSE 稳定中断。

2.22.170.2 接口定义

函数接口	void LL_RCC_DisableIT_LSERDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.171 LL_RCC_IsEnabledIT_LSERDY

2.22.171.1 功能介绍

检查是否使能 LSE 稳定中断。

2.22.171.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT_LSERDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.172 LL_RCC_EnableIT_MSIRDY

2.22.172.1 功能介绍

使能 MSI 稳定中断。

2.22.172.2 接口定义

函数接口	void LL_RCC_EnableIT_MSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.173 LL_RCC_DisableIT_MSIRDY

2.22.173.1 功能介绍

禁止使能 MSI 稳定中断。

2.22.173.2 接口定义

函数接口	void LL_RCC_DisableIT_MSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.174 LL_RCC_IsEnabledIT_MSIRDY

2.22.174.1 功能介绍

检查是否使能 MSI 稳定中断。

2.22.174.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT_MSIRDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.175 LL_RCC_EnableIT_HSIRDY

2.22.175.1 功能介绍

使能 HSI 稳定中断。

2.22.175.2 接口定义

函数接口	void LL_RCC_EnableIT_HSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.176 LL_RCC_DisableIT_HSIRDY

2.22.176.1 功能介绍

禁止使能 HSI 稳定中断。

2.22.176.2 接口定义

函数接口	void LL_RCC_DisableIT_HSIRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.177 LL_RCC_IsEnabledIT_HSIRDY

2.22.177.1 功能介绍

检查是否使能 HSI 稳定中断。

2.22.177.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT_HSIRDY (void)
输入	无
输出	无

返回值	{0,1}
资源使用	
说明	

2.22.178 LL_RCC_EnableIT_HSERDY

2.22.178.1 功能介绍

使能 HSE 稳定中断。

2.22.178.2 接口定义

函数接口	void LL_RCC_EnableIT_HSERDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.179 LL_RCC_DisableIT_HSERDY

2.22.179.1 功能介绍

禁止使能 HSE 稳定中断。

2.22.179.2 接口定义

函数接口	void LL_RCC_DisableIT_HSERDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.180 LL_RCC_IsEnabledIT_HSERDY

2.22.180.1 功能介绍

检查是否使能 HSE 稳定中断。

2.22.180.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT_HSERDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.181 LL_RCC_EnableIT_PLLRDY

2.22.181.1 功能介绍

使能 PLL 稳定中断。

2.22.181.2 接口定义

函数接口	void LL_RCC_EnableIT_PLLRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.182 LL_RCC_DisableIT_PLLRDY

2.22.182.1 功能介绍

禁止使能 PLL 稳定中断。

2.22.182.2 接口定义

函数接口	void LL_RCC_DisableIT_PLLRDY (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.22.183 LL_RCC_IsEnabledIT_PLLRDY

2.22.183.1 功能介绍

检查是否使能 PLL 稳定中断。

2.22.183.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT_PLLRDY (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.184 LL_RCC_EnableIT

2.22.184.1 功能介绍

使能中断。

2.22.184.2 接口定义

函数接口	void LL_RCC_EnableIT (uint32_t Flag)
输入	uint32_t Flag: { LL_RCC_FLAG_PLLRDY, LL_RCC_FLAG_HSERDY, LL_RCC_FLAG_HSIRDY, LL_RCC_FLAG_MSIRDY, LL_RCC_FLAG_LSERDY, LL_RCC_FLAG_LSIRDY }
输出	无
返回值	无

资源使用	
说明	

2.22.185 LL_RCC_DisableIT

2.22.185.1 功能介绍

禁止使能中断。

2.22.185.2 接口定义

函数接口	void LL_RCC_DisableIT (uint32_t Flag)
输入	uint32_t Flag: { LL_RCC_FLAG_PLLRDY, LL_RCC_FLAG_HSERDY, LL_RCC_FLAG_HSIRDY, LL_RCC_FLAG_MSIRDY, LL_RCC_FLAG_LSERDY, LL_RCC_FLAG_LSIRDY }
输出	无
返回值	无
资源使用	
说明	

2.22.186 LL_RCC_IsEnabledIT

2.22.186.1 功能介绍

检查是否使能中断。

2.22.186.2 接口定义

函数接口	uint32_t LL_RCC_IsEnabledIT (uint32_t Flag)
输入	uint32_t Flag: { LL_RCC_FLAG_PLLRDY, LL_RCC_FLAG_HSERDY, LL_RCC_FLAG_HSIRDY, LL_RCC_FLAG_MSIRDY, LL_RCC_FLAG_LSERDY, LL_RCC_FLAG_LSIRDY }
输出	无
返回值	{0,1}
资源使用	
说明	

2.22.187 LL_RCC_BGR_SetCalibTrimming

2.22.187.1 功能介绍

设置 BGR 微调值。

2.22.187.2 接口定义

函数接口	void LL_RCC_BGR_SetCalibTrimming (uint32_t Value)
输入	uint32_t Value: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.22.188 LL_RCC_BGR_GetCalibTrimming

2.22.188.1 功能介绍

获取 BGR 微调值。

2.22.188.2 接口定义

函数接口	uint32_t LL_RCC_BGR_GetCalibTrimming (void)
输入	无
输出	无
返回值	[0, 31]
资源使用	
说明	

2.22.189 LL_RCC_DAC_SetCalibTrimming

2.22.189.1 功能介绍

设置 DAC 微调值。

2.22.189.2 接口定义

函数接口	void LL_RCC_DAC_SetCalibTrimming (uint32_t DACIndex, uint32_t Value)
输入	uint32_t DACIndex: [0, 3] uint32_t Value: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.22.190 LL_RCC_DAC_GetCalibTrimming

2.22.190.1 功能介绍

获取 DAC 微调值。

2.22.190.2 接口定义

函数接口	uint32_t LL_RCC_DAC_GetCalibTrimming (uint32_t DACIndex)
输入	uint32_t DACIndex: [0, 3]
输出	无
返回值	[0, 31]
资源使用	
说明	

2.22.191 LL_RCC_OPA_SetCalibTrimming

2.22.191.1 功能介绍

设置 OPA 微调值。

2.22.191.2 接口定义

函数接口	void LL_RCC_OPA_SetCalibTrimming (uint32_t OPAIndex, uint32_t Value)
输入	uint32_t OPAIndex: [0, 3] uint32_t Value: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.22.192 LL_RCC_OPA_GetCalibTrimming

2.22.192.1 功能介绍

获取 OPA 微调值。

2.22.192.2 接口定义

函数接口	uint32_t LL_RCC_OPA_GetCalibTrimming (uint32_t OPAIndex)
输入	uint32_t OPAIndex: [0, 3]
输出	无
返回值	[0, 31]
资源使用	
说明	

2.22.193 LL_RCC_DeInit

2.22.193.1 功能介绍

将 RCC 时钟配置重置为默认重置状态。

2.22.193.2 接口定义

函数接口	ErrorStatus LL_RCC_DeInit (void)
输入	无
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.22.194 LL_RCC_GetSystemClocksFreq

2.22.194.1 功能介绍

获取系统时钟频率。

2.22.194.2 接口定义

函数接口	void LL_RCC_GetSystemClocksFreq (LL_RCC_ClocksTypeDef * RCC_Clocks)
输入	LL_RCC_ClocksTypeDef * RCC_Clocks

输出	无
返回值	无
资源使用	
说明	

2.22.195 LL_RCC_GetUSARTClockFreq

2.22.195.1 功能介绍

获取 USARTx 时钟频率。

2.22.195.2 接口定义

函数接口	uint32_t LL_RCC_GetUSARTClockFreq (uint32_t USARTxSource)
输入	uint32_t USARTxSource: { LL_RCC_USART1_CLKSOURCE }
输出	无
返回值	时钟频率
资源使用	
说明	

2.22.196 LL_RCC_GetI2CClockFreq

2.22.196.1 功能介绍

获取 I2Cx 时钟频率。

2.22.196.2 接口定义

函数接口	uint32_t LL_RCC_GetI2CClockFreq (uint32_t I2CxSource)
输入	uint32_t I2CxSource: { LL_RCC_I2C1_CLKSOURCE }
输出	无
返回值	时钟频率
资源使用	
说明	

2.22.197 LL_RCC_GetLPUARTClockFreq

2.22.197.1 功能介绍

获取 LPUARTx 时钟频率。

2.22.197.2 接口定义

函数接口	uint32_t LL_RCC_GetLPUARTClockFreq (uint32_t LPUARTxSource)
输入	uint32_t LPUARTxSource: { LL_RCC_LPUART1_CLKSOURCE }
输出	无
返回值	时钟频率
资源使用	
说明	

2.22.198 LL_RCC_GetLPTIMClockFreq

2.22.198.1 功能介绍

获取 LPTIM 时钟频率。

2.22.198.2 接口定义

函数接口	uint32_t LL_RCC_GetLPTIMClockFreq (uint32_t LPTIMxSource)
输入	uint32_t LPTIMxSource: { LL_RCC_LPTIM1_CLKSOURCE }
输出	无
返回值	时钟频率
资源使用	
说明	

2.22.199 LL_RCC_GetADCClockFreq

2.22.199.1 功能介绍

获取 ADCx 时钟频率。

2.22.199.2 接口定义

函数接口	uint32_t LL_RCC_GetADCClockFreq (uint32_t ADCxSource)
输入	uint32_t ADCxSource: { LL_RCC_ADC1_CLKSOURCE }
输出	无
返回值	时钟频率
资源使用	
说明	

2.22.200 LL_RCC_GetTIMClockFreq

2.22.200.1 功能介绍

获取 TIMx 时钟频率。

2.22.200.2 接口定义

函数接口	uint32_t LL_RCC_GetTIMClockFreq (uint32_t TIMxSource)
输入	uint32_t TIMxSource: { LL_RCC_TIM1_CLKSOURCE }
输出	无
返回值	时钟频率
资源使用	
说明	

2.22.201 LL_RCC_GetRTCClockFreq

2.22.201.1 功能介绍

获取 RTC 时钟频率。

2.22.201.2 接口定义

函数接口	uint32_t LL_RCC_GetRTCClockFreq (void)
------	--

输入	无
输出	无
返回值	时钟频率
资源使用	
说明	

2.23 SPI 模块

属性	类型	字段名	含义
SPI_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	FIFOC	FIFO 清空寄存器
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断标志清零寄存器
读写	uint32_t	DR	数据寄存器
LL_SPI_InitTypeDef			
读写	uint32_t	Mode	模式
读写	uint32_t	ClockPolarity	时钟极性
读写	uint32_t	ClockPhase	时钟相位
读写	uint32_t	NSS	NSS 模式
读写	uint32_t	BaudRate	波特率
读写	uint32_t	BitOrder	传输位序
读写	uint32_t	TxFIFODepth	发送 FIFO 阈值
读写	uint32_t	RxFIFODepth	接收 FIFO 阈值

2.23.1 LL_SPI_Enable

2.23.1.1 功能介绍

使能 SPI。

2.23.1.2 接口定义

函数接口	void LL_SPI_Enable (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.2 LL_SPI_Disable

2.23.2.1 功能介绍

禁止使能 SPI。

2.23.2.2 接口定义

函数接口	void LL_SPI_Disable (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.3 LL_SPI_IsEnabled

2.23.3.1 功能介绍

检查是否使能 SPI。

2.23.3.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabled (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.4 LL_SPI_SetMode

2.23.4.1 功能介绍

设置 SPI 主从模式。

2.23.4.2 接口定义

函数接口	void LL_SPI_SetMode (SPI_TypeDef * SPIx, uint32_t Mode)
输入	SPI_TypeDef * SPIx uint32_t Mode: { LL_SPI_MODE_MASTER, LL_SPI_MODE_SLAVE }
输出	无
返回值	无
资源使用	
说明	

2.23.5 LL_SPI_GetMode

2.23.5.1 功能介绍

获取 SPI 主从模式。

2.23.5.2 接口定义

函数接口	uint32_t LL_SPI_GetMode (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx

输出	无
返回值	{ LL_SPI_MODE_MASTER, LL_SPI_MODE_SLAVE }
资源使用	
说明	

2.23.6 LL_SPI_SetClockPhase

2.23.6.1 功能介绍

设置 SPI 时钟相位配置。

2.23.6.2 接口定义

函数接口	void LL_SPI_SetClockPhase (SPI_TypeDef * SPIx, uint32_t ClockPhase)
输入	SPI_TypeDef * SPIx uint32_t ClockPhase: { LL_SPI_PHASE_1EDGE, LL_SPI_PHASE_2EDGE }
输出	无
返回值	无
资源使用	
说明	

2.23.7 LL_SPI_GetClockPhase

2.23.7.1 功能介绍

获取 SPI 时钟相位配置。

2.23.7.2 接口定义

函数接口	uint32_t LL_SPI_GetClockPhase (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_PHASE_1EDGE, LL_SPI_PHASE_2EDGE }
资源使用	
说明	

2.23.8 LL_SPI_SetClockPolarity

2.23.8.1 功能介绍

设置 SPI 时钟极性。

2.23.8.2 接口定义

函数接口	void LL_SPI_SetClockPolarity (SPI_TypeDef * SPIx, uint32_t ClockPolarity)
输入	SPI_TypeDef * SPIx uint32_t ClockPolarity: { LL_SPI_POLARITY_LOW, LL_SPI_POLARITY_HIGH }
输出	无
返回值	无
资源使用	

说明	
----	--

2.23.9 LL_SPI_GetClockPolarity

2.23.9.1 功能介绍

获取 SPI 时钟极性。

2.23.9.2 接口定义

函数接口	uint32_t LL_SPI_GetClockPolarity (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_POLARITY_LOW, LL_SPI_POLARITY_HIGH }
资源使用	
说明	

2.23.10 LL_SPI_SetBaudRatePrescaler

2.23.10.1 功能介绍

设置 SPI 波特率控制。

2.23.10.2 接口定义

函数接口	void LL_SPI_SetBaudRatePrescaler (SPI_TypeDef * SPIx, uint32_t BaudRate)
输入	SPI_TypeDef * SPIx uint32_t BaudRate: { LL_SPI_BAUDRATEPRESCALER_DIV2, LL_SPI_BAUDRATEPRESCALER_DIV4, LL_SPI_BAUDRATEPRESCALER_DIV8, LL_SPI_BAUDRATEPRESCALER_DIV16, LL_SPI_BAUDRATEPRESCALER_DIV32, LL_SPI_BAUDRATEPRESCALER_DIV64, LL_SPI_BAUDRATEPRESCALER_DIV128 }
输出	无
返回值	无
资源使用	
说明	

2.23.11 LL_SPI_GetBaudRatePrescaler

2.23.11.1 功能介绍

获取 SPI 波特率控制。

2.23.11.2 接口定义

函数接口	uint32_t LL_SPI_GetBaudRatePrescaler (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_BAUDRATEPRESCALER_DIV2, LL_SPI_BAUDRATEPRESCALER_DIV4,

	LL_SPI_BAUDRATEPRESCALER_DIV8, LL_SPI_BAUDRATEPRESCALER_DIV16, LL_SPI_BAUDRATEPRESCALER_DIV32, LL_SPI_BAUDRATEPRESCALER_DIV64, LL_SPI_BAUDRATEPRESCALER_DIV128 }
资源使用	
说明	

2.23.12 LL_SPI_SetTransferBitOrder

2.23.12.1 功能介绍

设置 SPI 数据传输顺序。

2.23.12.2 接口定义

函数接口	void LL_SPI_SetTransferBitOrder (SPI_TypeDef * SPIx, uint32_t BitOrder)
输入	SPI_TypeDef * SPIx uint32_t BitOrder: { LL_SPI_LSB_FIRST, LL_SPI_MSB_FIRST }
输出	无
返回值	无
资源使用	
说明	

2.23.13 LL_SPI_GetTransferBitOrder

2.23.13.1 功能介绍

获取 SPI 数据传输顺序。

2.23.13.2 接口定义

函数接口	uint32_t LL_SPI_GetTransferBitOrder (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_LSB_FIRST, LL_SPI_MSB_FIRST }
资源使用	
说明	

2.23.14 LL_SPI_SetTxFIFOThreshold

2.23.14.1 功能介绍

设置 SPI 发送 FIFO 阈值配置。

2.23.14.2 接口定义

函数接口	void LL_SPI_SetTxFIFOThreshold (SPI_TypeDef * SPIx, uint32_t Threshold)
输入	SPI_TypeDef * SPIx uint32_t Threshold: { LL_SPI_FIFO_DEPTH_1_8, LL_SPI_FIFO_DEPTH_1_4, LL_SPI_FIFO_DEPTH_1_2, LL_SPI_FIFO_DEPTH_3_4,

	LL_SPI_FIFO_DEPTH_7_8, LL_SPI_FIFO_DEPTH_ALL }
输出	无
返回值	无
资源使用	
说明	

2.23.15 LL_SPI_GetTxFIFOThreshold

2.23.15.1 功能介绍

获取 SPI 发送 FIFO 阈值配置。

2.23.15.2 接口定义

函数接口	uint32_t LL_SPI_GetTxFIFOThreshold (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_FIFO_DEPTH_1_8, LL_SPI_FIFO_DEPTH_1_4, LL_SPI_FIFO_DEPTH_1_2, LL_SPI_FIFO_DEPTH_3_4, LL_SPI_FIFO_DEPTH_7_8, LL_SPI_FIFO_DEPTH_ALL }
资源使用	
说明	

2.23.16 LL_SPI_SetRxFIFOThreshold

2.23.16.1 功能介绍

设置 SPI 接收 FIFO 阈值配置。

2.23.16.2 接口定义

函数接口	void LL_SPI_SetRxFIFOThreshold (SPI_TypeDef * SPIx, uint32_t Threshold)
输入	SPI_TypeDef * SPIx uint32_t Threshold: { LL_SPI_FIFO_DEPTH_1_8, LL_SPI_FIFO_DEPTH_1_4, LL_SPI_FIFO_DEPTH_1_2, LL_SPI_FIFO_DEPTH_3_4, LL_SPI_FIFO_DEPTH_7_8, LL_SPI_FIFO_DEPTH_ALL }
输出	无
返回值	无
资源使用	
说明	

2.23.17 LL_SPI_GetRxFIFOThreshold

2.23.17.1 功能介绍

获取 SPI 接收 FIFO 阈值配置。

2.23.17.2 接口定义

函数接口	uint32_t LL_SPI_GetRxFIFOThreshold (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无

返回值	{ LL_SPI_FIFO_DEPTH_1_8, LL_SPI_FIFO_DEPTH_1_4, LL_SPI_FIFO_DEPTH_1_2, LL_SPI_FIFO_DEPTH_3_4, LL_SPI_FIFO_DEPTH_7_8, LL_SPI_FIFO_DEPTH_ALL }
资源使用	
说明	

2.23.18 LL_SPI_SetNSSMode

2.23.18.1 功能介绍

设置 SPI NSS 模式。

2.23.18.2 接口定义

函数接口	void LL_SPI_SetNSSMode (SPI_TypeDef * SPIx, uint32_t NSS)
输入	SPI_TypeDef * SPIx uint32_t NSS: { LL_SPI_NSS_HARD_INPUT LL_SPI_NSS_HARD_OUTPUT }
输出	无
返回值	无
资源使用	
说明	

2.23.19 LL_SPI_GetNSSMode

2.23.19.1 功能介绍

获取 SPI NSS 模式。

2.23.19.2 接口定义

函数接口	uint32_t LL_SPI_GetNSSMode (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_NSS_HARD_INPUT LL_SPI_NSS_HARD_OUTPUT }
资源使用	
说明	

2.23.20 LL_SPI_SetNSSOutputLevel

2.23.20.1 功能介绍

设置 SPI NSS 输出电平。

2.23.20.2 接口定义

函数接口	void LL_SPI_SetNSSOutputLevel (SPI_TypeDef * SPIx, uint32_t level)
输入	SPI_TypeDef * SPIx uint32_t level: { LL_SPI_NSS_OUTPUT_LOW, LL_SPI_NSS_OUTPUT_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.23.21 LL_SPI_GetNSSOutputLevel

2.23.21.1 功能介绍

获取 SPI NSS 输出电平。

2.23.21.2 接口定义

函数接口	uint32_t LL_SPI_GetNSSOutputLevel (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_NSS_OUTPUT_LOW, LL_SPI_NSS_OUTPUT_HIGH }
资源使用	
说明	

2.23.22 LL_SPI_SetSlaveNSSMode

2.23.22.1 功能介绍

设置 SPI NSS 的软件设置（只在 slave 模式有效）。

2.23.22.2 接口定义

函数接口	void LL_SPI_SetSlaveNSSMode (SPI_TypeDef * SPIx, uint32_t mode)
输入	SPI_TypeDef * SPIx uint32_t mode: { LL_SPI_SLAVE_NSS_SOFT, LL_SPI_SLAVE_NSS_PIN }
输出	无
返回值	无
资源使用	
说明	

2.23.23 LL_SPI_GetSlaveNSSMode

2.23.23.1 功能介绍

获取 SPI NSS 的软件设置（只在 slave 模式有效）。

2.23.23.2 接口定义

函数接口	uint32_t LL_SPI_GetSlaveNSSMode (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{ LL_SPI_SLAVE_NSS_SOFT, LL_SPI_SLAVE_NSS_PIN }
资源使用	
说明	

2.23.24 LL_SPI_IsActiveFlag_TXFT

2.23.24.1 功能介绍

检查发送 FIFO 阈值标志。

2.23.24.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_TXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx

输出	无
返回值	{0,1}
资源使用	
说明	

2.23.25 LL_SPI_IsActiveFlag_TXFNF

2.23.25.1 功能介绍

检查发送 FIFO 非满标志。

2.23.25.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_TXFNF(SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.26 LL_SPI_IsActiveFlag_TXFE

2.23.26.1 功能介绍

检查发送 FIFO 空标志。

2.23.26.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_TXFE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.27 LL_SPI_IsActiveFlag_RXFT

2.23.27.1 功能介绍

检查接收 FIFO 阈值标志。

2.23.27.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_RXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.28 LL_SPI_IsActiveFlag_RXFNE

2.23.28.1 功能介绍

检查接收 FIFO 非空标志。

2.23.28.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_RXFNE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.29 LL_SPI_IsActiveFlag_RXFF

2.23.29.1 功能介绍

检查接收 FIFO 已满标志。

2.23.29.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_RXFF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.30 LL_SPI_IsActiveFlag_MMF

2.23.30.1 功能介绍

检查主机模式冲突标志。

2.23.30.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_MMF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.31 LL_SPI_IsActiveFlag_SME

2.23.31.1 功能介绍

检查从机模式 NSS 状态标志。

2.23.31.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_SME (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.32 LL_SPI_IsActiveFlag_OVR

2.23.32.1 功能介绍

检查上溢标志。

2.23.32.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_OVR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.33 LL_SPI_IsActiveFlag_BSY

2.23.33.1 功能介绍

检查 SPI 总线传输状态标志。

2.23.33.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_BSY (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.34 LL_SPI_IsActiveFlag_UDR

2.23.34.1 功能介绍

检查从机模式下溢标志。

2.23.34.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag_UDR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.35 LL_SPI_IsActiveFlag

2.23.35.1 功能介绍

检查标志位。

2.23.35.2 接口定义

函数接口	uint32_t LL_SPI_IsActiveFlag (SPI_TypeDef * SPIx, uint32_t Flag)
输入	SPI_TypeDef * SPIx uint32_t Flag: { LL_SPI_ISR_TXFE, LL_SPI_ISR_RXFT, LL_SPI_ISR_RXFNE,

	LL_SPI_ISR_RXFF, LL_SPI_ISR_BSY, LL_SPI_ISR_UDR, LL_SPI_ISR_SME, LL_SPI_ISR_OVR, LL_SPI_ISR_MMF }
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.36 LL_SPI_ClearFlag_MMF

2.23.36.1 功能介绍

清除主机模式冲突标志。

2.23.36.2 接口定义

函数接口	void LL_SPI_ClearFlag_MMF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.37 LL_SPI_ClearFlag_SME

2.23.37.1 功能介绍

清除从机模式 NSS 状态标志。

2.23.37.2 接口定义

函数接口	void LL_SPI_ClearFlag_SME (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.38 LL_SPI_ClearFlag_OVR

2.23.38.1 功能介绍

清除上溢标志。

2.23.38.2 接口定义

函数接口	void LL_SPI_ClearFlag_OVR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.39 LL_SPI_ClearFlag

2.23.39.1 功能介绍

清除标志位。

2.23.39.2 接口定义

函数接口	void LL_SPI_ClearFlag (SPI_TypeDef * SPIx, uint32_t Flag)
输入	SPI_TypeDef * SPIx uint32_t Flag: { LL_SPI_ISR_UDR, LL_SPI_ISR_SME, LL_SPI_ISR_OVR, LL_SPI_ISR_MMF }
输出	无
返回值	无
资源使用	
说明	

2.23.40 LL_SPI_ClearFlag_UDR

2.23.40.1 功能介绍

清除从机模式下溢标志。

2.23.40.2 接口定义

函数接口	void LL_SPI_ClearFlag_UDR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.41 LL_SPI_EnableIT_MMF

2.23.41.1 功能介绍

使能 MMF 错误中断。

2.23.41.2 接口定义

函数接口	void LL_SPI_EnableIT_MMF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.42 LL_SPI_EnableIT_OVR

2.23.42.1 功能介绍

使能上溢错误中断。

2.23.42.2 接口定义

函数接口	void LL_SPI_EnableIT_OVR (SPI_TypeDef * SPIx)
------	---

输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.43LL_SPI_EnableIT_UDR

2.23.43.1 功能介绍

使能下溢错误中断。

2.23.43.2 接口定义

函数接口	void LL_SPI_EnableIT_UDR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.44LL_SPI_EnableIT_SME

2.23.44.1 功能介绍

使能从模式 NSS 状态中断。

2.23.44.2 接口定义

函数接口	void LL_SPI_EnableIT_SME (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.45LL_SPI_EnableIT_RXFF

2.23.45.1 功能介绍

使能接收 FIFO 为满中断。

2.23.45.2 接口定义

函数接口	void LL_SPI_EnableIT_RXFF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.46 LL_SPI_EnableIT_RXNE

2.23.46.1 功能介绍

使能接收 FIFO 非空中断。

2.23.46.2 接口定义

函数接口	void LL_SPI_EnableIT_RXNE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.47 LL_SPI_EnableIT_RXFT

2.23.47.1 功能介绍

使能接收 FIFO 阈值中断。

2.23.47.2 接口定义

函数接口	void LL_SPI_EnableIT_RXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.48 LL_SPI_EnableIT_TXFT

2.23.48.1 功能介绍

使能发送 FIFO 阈值中断。

2.23.48.2 接口定义

函数接口	void LL_SPI_EnableIT_TXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.49 LL_SPI_EnableIT_TXFNF

2.23.49.1 功能介绍

使能 MMF 发送 FIFO 非满中断。

2.23.49.2 接口定义

函数接口	void LL_SPI_EnableIT_TXFNF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无

返回值	无
资源使用	
说明	

2.23.50LL_SPI_EnableIT_TXFE

2.23.50.1 功能介绍

使能发送 FIFO 为空中断。

2.23.50.2 接口定义

函数接口	void LL_SPI_EnableIT_TXFE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.51LL_SPI_EnableIT

2.23.51.1 功能介绍

使能中断。

2.23.51.2 接口定义

函数接口	void LL_SPI_EnableIT (SPI_TypeDef * SPIx, uint32_t Interrupt)
输入	SPI_TypeDef * SPIx uint32_t Interrupt: { LL_SPI_TX_FTIE, LL_SPI_TX_FNFIE, LL_SPI_TX_FEIE, LL_SPI_RX_FTIE, LL_SPI_RX_FNEIE, LL_SPI_RX_FFIE, LL_SPI_MMF_IE, LL_SPI_OVR_IE, LL_SPI_UDR_IE, LL_SPI_SME_IE }
输出	无
返回值	无
资源使用	
说明	

2.23.52LL_SPI_DisableIT_MMF

2.23.52.1 功能介绍

禁止使能 MMF 错误中断。

2.23.52.2 接口定义

函数接口	void LL_SPI_DisableIT_MMF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.53 LL_SPI_DisableIT_OVR

2.23.53.1 功能介绍

禁止使能上溢错误中断。

2.23.53.2 接口定义

函数接口	void LL_SPI_DisableIT_OVR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.54 LL_SPI_DisableIT_UDR

2.23.54.1 功能介绍

禁止使能下溢错误中断。

2.23.54.2 接口定义

函数接口	void LL_SPI_DisableIT_UDR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.55 LL_SPI_DisableIT_SME

2.23.55.1 功能介绍

禁止使能从模式 NSS 状态中断。

2.23.55.2 接口定义

函数接口	void LL_SPI_DisableIT_SME (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.56 LL_SPI_DisableIT_RXFF

2.23.56.1 功能介绍

禁止使能接收 FIFO 为满中断。

2.23.56.2 接口定义

函数接口	void LL_SPI_DisableIT_RXFF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无

返回值	无
资源使用	
说明	

2.23.57 LL_SPI_DisableIT_RXNE

2.23.57.1 功能介绍

禁止使能接收 FIFO 非空中断。

2.23.57.2 接口定义

函数接口	void LL_SPI_DisableIT_RXNE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.58 LL_SPI_DisableIT_RXFT

2.23.58.1 功能介绍

禁止使能接收 FIFO 阈值中断。

2.23.58.2 接口定义

函数接口	void LL_SPI_DisableIT_RXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.59 LL_SPI_DisableIT_TXFT

2.23.59.1 功能介绍

禁止使能发送 FIFO 阈值中断。

2.23.59.2 接口定义

函数接口	void LL_SPI_DisableIT_TXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.60 LL_SPI_DisableIT_TXFNF

2.23.60.1 功能介绍

禁止使能发送 FIFO 非满中断。

2.23.60.2 接口定义

函数接口	void LL_SPI_DisableIT_TXFNF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.61 LL_SPI_DisableIT_TXFE

2.23.61.1 功能介绍

禁止使能发送 FIFO 为空中断。

2.23.61.2 接口定义

函数接口	void LL_SPI_DisableIT_TXFE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.62 LL_SPI_DisableIT

2.23.62.1 功能介绍

禁止使能中断。

2.23.62.2 接口定义

函数接口	void LL_SPI_DisableIT (SPI_TypeDef * SPIx, uint32_t Interrupt)
输入	SPI_TypeDef * SPIx uint32_t Interrupt: { LL_SPI_TX_FTIE, LL_SPI_TX_FNFIE, LL_SPI_TX_FEIE, LL_SPI_RX_FTIE, LL_SPI_RX_FNEIE, LL_SPI_RX_FFIE, LL_SPI_MMF_IE, LL_SPI_OVR_IE, LL_SPI_UDR_IE, LL_SPI_SME_IE }
输出	无
返回值	无
资源使用	
说明	

2.23.63 LL_SPI_IsEnabledIT_MMF

2.23.63.1 功能介绍

检查是否使能 MMF 错误中断。

2.23.63.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_MMF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx

输出	无
返回值	{0,1}
资源使用	
说明	

2.23.64 LL_SPI_IsEnabledIT_OVR

2.23.64.1 功能介绍

检查是否使能上溢错误中断。

2.23.64.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_OVR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.65 LL_SPI_IsEnabledIT_UDR

2.23.65.1 功能介绍

检查是否使能下溢错误中断。

2.23.65.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_UDR (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.66 LL_SPI_IsEnabledIT_SME

2.23.66.1 功能介绍

检查是否使能从模式 NSS 状态中断。

2.23.66.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_SME (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.67 LL_SPI_IsEnabledIT_RXFF

2.23.67.1 功能介绍

检查是否使能接收 FIFO 为满中断。

2.23.67.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_RXFF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.68 LL_SPI_IsEnabledIT_RXFNE

2.23.68.1 功能介绍

检查是否使能接收 FIFO 非空中断。

2.23.68.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_RXFNE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.69 LL_SPI_IsEnabledIT_RXFT

2.23.69.1 功能介绍

检查是否使能接收 FIFO 阈值中断。

2.23.69.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_RXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.70 LL_SPI_IsEnabledIT_TXFE

2.23.70.1 功能介绍

检查是否使能发送 FIFO 为空中断。

2.23.70.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_TXFE (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.71 LL_SPI_IsEnabledIT_TXFNF

2.23.71.1 功能介绍

检查是否使能发送 FIFO 非满中断。

2.23.71.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_TXFNF (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.72 LL_SPI_IsEnabledIT_TXFT

2.23.72.1 功能介绍

检查是否使能发送 FIFO 阈值中断。

2.23.72.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT_TXFT (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.73 LL_SPI_IsEnabledIT

2.23.73.1 功能介绍

检查是否使能中断。

2.23.73.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledIT (SPI_TypeDef * SPIx, uint32_t Interrupt)
输入	SPI_TypeDef * SPIx uint32_t Interrupt: { LL_SPI_TX_FTIE, LL_SPI_TX_FNFIE, LL_SPI_TX_FEIE, LL_SPI_RX_FTIE, LL_SPI_RX_FNEIE, LL_SPI_RX_FFIE, LL_SPI_MMF_IE, LL_SPI_OVR_IE, LL_SPI_UDR_IE, LL_SPI_SME_IE }
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.74 LL_SPI_EnableDMAReq_RX

2.23.74.1 功能介绍

使能 DMA 方式接收。

2.23.74.2 接口定义

函数接口	void LL_SPI_EnableDMAReq_RX (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.75 LL_SPI_DisableDMAReq_RX

2.23.75.1 功能介绍

禁止使能 DMA 方式接收。

2.23.75.2 接口定义

函数接口	void LL_SPI_DisableDMAReq_RX (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.76 LL_SPI_IsEnabledDMAReq_RX

2.23.76.1 功能介绍

检查是否使能 DMA 方式接收。

2.23.76.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledDMAReq_RX (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.77 LL_SPI_EnableDMAReq_TX

2.23.77.1 功能介绍

使能 DMA 方式发送。

2.23.77.2 接口定义

函数接口	void LL_SPI_EnableDMAReq_TX (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.78 LL_SPI_DisableDMAReq_TX

2.23.78.1 功能介绍

禁止使能 DMA 方式发送。

2.23.78.2 接口定义

函数接口	void LL_SPI_DisableDMAReq_TX (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.79 LL_SPI_IsEnabledDMAReq_TX

2.23.79.1 功能介绍

检查是否使能 DMA 方式发送。

2.23.79.2 接口定义

函数接口	uint32_t LL_SPI_IsEnabledDMAReq_TX (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	{0,1}
资源使用	
说明	

2.23.80 LL_SPI_DMA_GetRegAddr

2.23.80.1 功能介绍

获取用于 DMA 传输的数据寄存器地址。

2.23.80.2 接口定义

函数接口	uint32_t LL_SPI_DMA_GetRegAddr (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.23.81 LL_SPI_ClearTransformFIFOData

2.23.81.1 功能介绍

清除发送 FIFO 数据。

2.23.81.2 接口定义

函数接口	void LL_SPI_ClearTransformFIFOData (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无

返回值	无
资源使用	
说明	

2.23.82 LL_SPI_ClearReceiveFIFOData

2.23.82.1 功能介绍

清除接收 FIFO 数据。

2.23.82.2 接口定义

函数接口	void LL_SPI_ClearReceiveFIFOData (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	无
资源使用	
说明	

2.23.83 LL_SPI_ReceiveData8

2.23.83.1 功能介绍

读取数据寄存器的 8 位。

2.23.83.2 接口定义

函数接口	uint8_t LL_SPI_ReceiveData8 (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.23.84 LL_SPI_TransmitData8

2.23.84.1 功能介绍

写入数据寄存器的 8 位。

2.23.84.2 接口定义

函数接口	void LL_SPI_TransmitData8 (SPI_TypeDef * SPIx, uint8_t TxData)
输入	SPI_TypeDef * SPIx uint8_t TxData: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.23.85 LL_SPI_DeInit

2.23.85.1 功能介绍

清除 SPI 初始化参数。

2.23.85.2 接口定义

函数接口	ErrorStatus LL_SPI_DeInit (SPI_TypeDef * SPIx)
输入	SPI_TypeDef * SPIx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.23.86 LL_SPI_Init

2.23.86.1 功能介绍

SPI 初始化。

2.23.86.2 接口定义

函数接口	ErrorStatus LL_SPI_Init (SPI_TypeDef * SPIx, LL_SPI_InitTypeDef * SPI_InitStruct)
输入	SPI_TypeDef * SPIx LL_SPI_InitTypeDef * SPI_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.23.87 LL_SPI_StructInit

2.23.87.1 功能介绍

SPI 结构体初始化。

2.23.87.2 接口定义

函数接口	void LL_SPI_StructInit (LL_SPI_InitTypeDef * SPI_InitStruct)
输入	LL_SPI_InitTypeDef * SPI_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.24 SYSTEM 模块

2.24.1 LL_SYSCFG_SetRemapMemory

2.24.1.1 功能介绍

设置存储器映射到地址 0x0000 0000。

2.24.1.2 接口定义

函数接口	void LL_SYSCFG_SetRemapMemory (uint32_t MemoryMode)
输入	uint32_t MemoryMode: { LL_SYSCFG_REMAP_FLASH,

	LL_SYSCFG_REMAP_SYSTEMFLASH }
输出	无
返回值	无
资源使用	
说明	

2.24.2 LL_SYSCFG_GetRemapMemory

2.24.2.1 功能介绍

获取映射到地址 0x0000 0000 的存储器。

2.24.2.2 接口定义

函数接口	uint32_t LL_SYSCFG_GetRemapMemory (void)
输入	无
输出	无
返回值	{ LL_SYSCFG_REMAP_FLASH, LL_SYSCFG_REMAP_SYSTEMFLASH }
资源使用	
说明	

2.24.3 LL_SYSCFG_SetIRModEnvelopeSignal

2.24.3.1 功能介绍

设置 IRTIM 调制信号。

2.24.3.2 接口定义

函数接口	void LL_SYSCFG_SetIRModEnvelopeSignal (uint32_t IRMSource)
输入	uint32_t IRMSource: { LL_SYSCFG_IR_MOD_TIM4, LL_SYSCFG_IR_MOD_USART2, LL_SYSCFG_IR_MOD_USART3 }
输出	无
返回值	无
资源使用	
说明	

2.24.4 LL_SYSCFG_GetIRModEnvelopeSignal

2.24.4.1 功能介绍

获取 IRTIM 调制信号。

2.24.4.2 接口定义

函数接口	uint32_t LL_SYSCFG_GetIRModEnvelopeSignal (void)
输入	无
输出	无
返回值	{ LL_SYSCFG_IR_MOD_TIM4, LL_SYSCFG_IR_MOD_USART2, LL_SYSCFG_IR_MOD_USART3 }
资源使用	

说明	
----	--

2.24.5 LL_SYSCFG_SetIRPolarity

2.24.5.1 功能介绍

设置 IR_OUT 输出信号极性。

2.24.5.2 接口定义

函数接口	void LL_SYSCFG_SetIRPolarity (uint32_t IRPolarity)
输入	uint32_t IRPolarity: { LL_SYSCFG_IR_POL_INVERTED, LL_SYSCFG_IR_POL_NOT_INVERTED }
输出	无
返回值	无
资源使用	
说明	

2.24.6 LL_SYSCFG_GetIRPolarity

2.24.6.1 功能介绍

获取 IR_OUT 输出信号极性。

2.24.6.2 接口定义

函数接口	uint32_t LL_SYSCFG_GetIRPolarity (void)
输入	无
输出	无
返回值	{ LL_SYSCFG_IR_POL_INVERTED, LL_SYSCFG_IR_POL_NOT_INVERTED }
资源使用	
说明	

2.24.7 LL_SYSCFG_SetTIMBreakInputs

2.24.7.1 功能介绍

设置连接到 TIM1 中断输入。

2.24.7.2 接口定义

函数接口	void LL_SYSCFG_SetTIMBreakInputs (uint32_t TIMBreak)
输入	uint32_t TIMBreak: { LL_SYSCFG_TIMBREAK_PVD, LL_SYSCFG_TIMBREAK_LOCKUP }
输出	无
返回值	无
资源使用	
说明	

2.24.8 LL_SYSCFG_GetTIMBreakInputs

2.24.8.1 功能介绍

获取连接到 TIM1 中断输入。

2.24.8.2 接口定义

函数接口	uint32_t LL_SYSCFG_GetTIMBreakInputs (void)
输入	无
输出	无
返回值	{ LL_SYSCFG_TIMBREAK_PVD, LL_SYSCFG_TIMBREAK_LOCKUP }
资源使用	
说明	

2.24.9 LL_DBGMCU_WriteData

2.24.9.1 功能介绍

写入调试模块数据。

2.24.9.2 接口定义

函数接口	void LL_DBGMCU_WriteData (uint32_t Data)
输入	uint32_t Data: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.24.10 LL_DBGMCU_ReadData

2.24.10.1 功能介绍

读出调试模块数据。

2.24.10.2 接口定义

函数接口	uint32_t LL_DBGMCU_ReadData (void)
输入	无
输出	无
返回值	[0x00000000, 0xFFFFFFFF]
资源使用	
说明	

2.24.11 LL_DBGMCU_EnableDBGStopMode

2.24.11.1 功能介绍

使能在 Stop 模式调试 MCU 功能。

2.24.11.2 接口定义

函数接口	void LL_DBGMCU_EnableDBGStopMode (void)
输入	无
输出	无
返回值	无
资源使用	

说明	
----	--

2.24.12 LL_DBGMCU_DisableDBGStopMode

2.24.12.1 功能介绍

禁止使能在 Stop 模式调试 MCU 功能。

2.24.12.2 接口定义

函数接口	void LL_DBGMCU_DisableDBGStopMode (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

2.24.13 LL_DBGMCU_IsEnabledDBGStopMode

2.24.13.1 功能介绍

检查是否使能在 Stop 模式调试 MCU 功能。

2.24.13.2 接口定义

函数接口	uint32_t LL_DBGMCU_IsEnabledDBGStopMode (void)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.24.14 LL_DBGMCU_APB1_GRP1_FreezePeriph

2.24.14.1 功能介绍

内核停止时， APB1 外设停止计数功能。

2.24.14.2 接口定义

函数接口	void LL_DBGMCU_APB1_GRP1_FreezePeriph (uint32_t Periphs)
输入	uint32_t Periphs: { LL_DBGMCU_APB1_GRP1_TIM2_STOP, LL_DBGMCU_APB1_GRP1_TIM3_STOP, LL_DBGMCU_APB1_GRP1_TIM4_STOP, LL_DBGMCU_APB1_GRP1_TIM5_STOP, LL_DBGMCU_APB1_GRP1_TIM6_STOP, LL_DBGMCU_APB1_GRP1_TIM7_STOP, LL_DBGMCU_APB1_GRP1_RTC_STOP, LL_DBGMCU_APB1_GRP1_WWDG_STOP, LL_DBGMCU_APB1_GRP1_IWDG_STOP, LL_DBGMCU_APB1_GRP1_LPTIM_STOP }
输出	无
返回值	无

资源使用	
说明	

2.24.15 LL_DBGMCU_APB1_GRP1_UnFreezePeriph

2.24.15.1 功能介绍

内核停止时， APB1 外设正常计数功能。

2.24.15.2 接口定义

函数接口	void LL_DBGMCU_APB1_GRP1_UnFreezePeriph (uint32_t Periphs)
输入	uint32_t Periphs: { LL_DBGMCU_APB1_GRP1_TIM2_STOP, LL_DBGMCU_APB1_GRP1_TIM3_STOP, LL_DBGMCU_APB1_GRP1_TIM4_STOP, LL_DBGMCU_APB1_GRP1_TIM5_STOP, LL_DBGMCU_APB1_GRP1_TIM6_STOP, LL_DBGMCU_APB1_GRP1_TIM7_STOP, LL_DBGMCU_APB1_GRP1_RTC_STOP, LL_DBGMCU_APB1_GRP1_WWDG_STOP, LL_DBGMCU_APB1_GRP1_IWDG_STOP, LL_DBGMCU_APB1_GRP1_LPTIM_STOP }
输出	无
返回值	无
资源使用	
说明	

2.24.16 LL_DBGMCU_APB1_GRP1_IsFreezedPeriph

2.24.16.1 功能介绍

内核停止时， APB1 外设是否停止计数功能。

2.24.16.2 接口定义

函数接口	uint32_t LL_DBGMCU_APB1_GRP1_IsFreezedPeriph (uint32_t Periphs)
输入	uint32_t Periphs: { LL_DBGMCU_APB1_GRP1_TIM2_STOP, LL_DBGMCU_APB1_GRP1_TIM3_STOP, LL_DBGMCU_APB1_GRP1_TIM4_STOP, LL_DBGMCU_APB1_GRP1_TIM5_STOP, LL_DBGMCU_APB1_GRP1_TIM6_STOP, LL_DBGMCU_APB1_GRP1_TIM7_STOP, LL_DBGMCU_APB1_GRP1_RTC_STOP, LL_DBGMCU_APB1_GRP1_WWDG_STOP, LL_DBGMCU_APB1_GRP1_IWDG_STOP, LL_DBGMCU_APB1_GRP1_LPTIM_STOP }
输出	无
返回值	{0,1}

资源使用	
说明	

2.24.17 LL_DBGMCU_APB2_GRP1_FreezePeriph

2.24.17.1 功能介绍

内核停止时， APB2 外设停止计数功能。

2.24.17.2 接口定义

函数接口	void LL_DBGMCU_APB2_GRP1_FreezePeriph (uint32_t Periphs)
输入	uint32_t Periphs: { LL_DBGMCU_APB2_GRP1_TIM1_STOP }
输出	无
返回值	无
资源使用	
说明	

2.24.18 LL_DBGMCU_APB2_GRP1_UnFreezePeriph

2.24.18.1 功能介绍

内核停止时， APB2 外设正常计数功能。

2.24.18.2 接口定义

函数接口	void LL_DBGMCU_APB2_GRP1_UnFreezePeriph (uint32_t Periphs)
输入	uint32_t Periphs: { LL_DBGMCU_APB2_GRP1_TIM1_STOP }
输出	无
返回值	无
资源使用	
说明	

2.24.19 LL_DBGMCU_APB2_GRP1_IsFreezedPeriph

2.24.19.1 功能介绍

内核停止时， APB2 外设是否停止计数功能。

2.24.19.2 接口定义

函数接口	uint32_t LL_DBGMCU_APB2_GRP1_IsFreezedPeriph (uint32_t Periphs)
输入	uint32_t Periphs: { LL_DBGMCU_APB2_GRP1_TIM1_STOP }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25 TIM 模块

属性	类型	字段名	含义
TIM_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	SMCR	从模式控制寄存器
读写	uint32_t	DIER	DMA/ 中断使能寄存器
读写	uint32_t	SR	状态寄存器
读写	uint32_t	EGR	事件产生寄存器
读写	uint32_t	CCMR1_Output	捕获/ 比较模式寄存器 1 输出比较模式
读写	uint32_t	CCMR1_Input	捕获/ 比较模式寄存器 1 输入捕获模式
读写	uint32_t	CCMR2_Output	捕获/ 比较模式寄存器 2 输出比较模式
读写	uint32_t	CCMR2_Input	捕获/ 比较模式寄存器 2 输入捕获模式
读写	uint32_t	CCER	捕获/ 比较使能寄存器
读写	uint32_t	CNT	计数值寄存器
读写	uint32_t	PSC	预分频器寄存器
读写	uint32_t	ARR	自动重载值寄存器
读写	uint32_t	RCR	重复计数器寄存器
读写	uint32_t	CCR1	捕获/ 比较寄存器 1
读写	uint32_t	CCR2	捕获/ 比较寄存器 2
读写	uint32_t	CCR3	捕获/ 比较寄存器 3
读写	uint32_t	CCR4	捕获/ 比较寄存器 4
读写	uint32_t	BDTR	断路和死区寄存器
读写	uint32_t	DCR	DMA 控制寄存器
读写	uint32_t	DMAR	DMA 全传输地址寄存器
读写	uint32_t	OR1	配置 寄存器
读写	uint32_t	CCMR3_Output	捕获/ 比较模式寄

			寄存器 3
读写	uint32_t	CCR5	捕获/ 比较寄存器 5
读写	uint32_t	CCR6	捕获/ 比较寄存器 6
读写	uint32_t	AF1	轮换功能寄存器 1
读写	uint32_t	AF2	轮换功能寄存器 3
读写	uint32_t	TISEL	定时器输入选择寄存器
LL_TIM_InitTypeDef			
读写	uint32_t	Prescaler	预分频器值
读写	uint32_t	CounterMode	计数模式
读写	uint32_t	ClockDivision	时钟分频
读写	uint32_t	RepetitionCounter	重复计数器值
LL_TIM_OC_InitTypeDef			
读写	uint32_t	OCMode	输出比较模式
读写	uint32_t	OCState	输出比较状态
读写	uint32_t	OCNState	互补输出比较状态
读写	uint32_t	CompareValue	比较值
读写	uint32_t	OCPolarity	输出极性
读写	uint32_t	OCNPolarity	互补输出极性
读写	uint32_t	OCIdleState	输出空闲状态
读写	uint32_t	OCNIdleState	互补输出空闲状态
LL_TIM_IC_InitTypeDef			
读写	uint32_t	ICPolarity	输入极性
读写	uint32_t	ICActiveInput	通道方向（输入/输出）以及所使用的输入
读写	uint32_t	ICPrescaler	输入捕获预分频器
读写	uint32_t	ICFilter	输入捕获滤波器
LL_TIM_ENCODER_InitTypeDef			
读写	uint32_t	EncoderMode	从模式选择
读写	uint32_t	IC1Polarity	输入 1 极性
读写	uint32_t	IC1ActiveInput	输入 1 通道方向（输入/输出）以及所使用的输入
读写	uint32_t	IC1Prescaler	输入 1 捕获预分频器

读写	uint32_t	IC1Filter	输入 1 捕获滤波器
读写	uint32_t	IC2Polarity	输入 2 极性
读写	uint32_t	IC2ActiveInput	输入 2 通道方向（输入/输出）以及所使用的输入
读写	uint32_t	IC2Prescaler	输入 2 捕获预分频器
读写	uint32_t	IC2Filter	输入 2 捕获滤波器
LL_TIM_HALLSENSOR_InitTypeDef			
读写	uint32_t	IC1Polarity	输入极性
读写	uint32_t	IC1Prescaler	输入捕获预分频器
读写	uint32_t	IC1Filter	输入捕获滤波器
读写	uint32_t	CommutationDelay	捕获/比较值
LL_TIM_BDTR_InitTypeDef			
读写	uint32_t	OSSRState	运行模式下的关闭状态选择
读写	uint32_t	OSSIState	空闲模式下的关闭状态选择
读写	uint32_t	LockLevel	锁定级别
读写	uint32_t	DeadTime	死区发生时间
读写	uint32_t	BreakState	断路状态
读写	uint32_t	BreakPolarity	断路极性
读写	uint32_t	BreakFilter	断路滤波
读写	uint32_t	BreakAFMode	断路复用模式
读写	uint32_t	Break2State	断路 2 状态
读写	uint32_t	Break2Polarity	断路 2 极性
读写	uint32_t	Break2Filter	断路 2 滤波
读写	uint32_t	Break2AFMode	断路 2 复用模式
读写	uint32_t	AutomaticOutput	自动输出

2.25.1 LL_TIM_EnableCounter

2.25.1.1 功能介绍

使能 TIM 计数。

2.25.1.2 接口定义

函数接口	void LL_TIM_EnableCounter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx

输出	无
返回值	无
资源使用	
说明	

2.25.2 LL_TIM_DisableCounter

2.25.2.1 功能介绍

禁止使能 TIM 计数。

2.25.2.2 接口定义

函数接口	void LL_TIM_DisableCounter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.3 LL_TIM_IsEnabledCounter

2.25.3.1 功能介绍

检查是否使能 TIM 计数。

2.25.3.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledCounter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.4 LL_TIM_EnableUpdateEvent

2.25.4.1 功能介绍

启用更新事件生成。

2.25.4.2 接口定义

函数接口	void LL_TIM_EnableUpdateEvent (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.5 LL_TIM_DisableUpdateEvent

2.25.5.1 功能介绍

禁止启用更新事件生成。

2.25.5.2 接口定义

函数接口	void LL_TIM_DisableUpdateEvent (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.6 LL_TIM_IsEnabledUpdateEvent

2.25.6.1 功能介绍

启用更新事件生成。

2.25.6.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledUpdateEvent (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.7 LL_TIM_SetUpdateSource

2.25.7.1 功能介绍

设置更新事件源。

2.25.7.2 接口定义

函数接口	void LL_TIM_SetUpdateSource (TIM_TypeDef * TIMx, uint32_t UpdateSource)
输入	TIM_TypeDef * TIMx uint32_t UpdateSource: { LL_TIM_UPDATESOURCE_REGULAR, LL_TIM_UPDATESOURCE_COUNTER }
输出	无
返回值	无
资源使用	
说明	

2.25.8 LL_TIM_GetUpdateSource

2.25.8.1 功能介绍

获取更新事件源。

2.25.8.2 接口定义

函数接口	uint32_t LL_TIM_GetUpdateSource (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	{ LL_TIM_UPDATESOURCE_REGULAR, LL_TIM_UPDATESOURCE_COUNTER }
资源使用	
说明	

2.25.9 LL_TIM_SetOnePulseMode

2.25.9.1 功能介绍

设置单脉冲模式。

2.25.9.2 接口定义

函数接口	void LL_TIM_SetOnePulseMode (TIM_TypeDef * TIMx, uint32_t OnePulseMode)
输入	TIM_TypeDef * TIMx uint32_t OnePulseMode: { LL_TIM_ONEPULSEMODE_SINGLE, LL_TIM_ONEPULSEMODE_REPETITIVE }
输出	无
返回值	无
资源使用	
说明	

2.25.10 LL_TIM_GetOnePulseMode

2.25.10.1 功能介绍

获取单脉冲模式。

2.25.10.2 接口定义

函数接口	uint32_t LL_TIM_GetOnePulseMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_ONEPULSEMODE_SINGLE, LL_TIM_ONEPULSEMODE_REPETITIVE }
资源使用	
说明	

2.25.11 LL_TIM_SetCounterMode

2.25.11.1 功能介绍

设置定时器计数器计数模式。

2.25.11.2 接口定义

函数接口	void LL_TIM_SetCounterMode (TIM_TypeDef * TIMx, uint32_t CounterMode)
输入	TIM_TypeDef * TIMx uint32_t CounterMode: { LL_TIM_COUNTERMODE_UP, LL_TIM_COUNTERMODE_DOWN, LL_TIM_COUNTERMODE_CENTER_UP, LL_TIM_COUNTERMODE_CENTER_DOWN,

	LL_TIM_COUNTERMODE_CENTER_UP_DOWN }
输出	无
返回值	无
资源使用	
说明	

2.25.12 LL_TIM_GetCounterMode

2.25.12.1 功能介绍

获取定时器计数器计数模式。

2.25.12.2 接口定义

函数接口	uint32_t LL_TIM_GetCounterMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_COUNTERMODE_UP, LL_TIM_COUNTERMODE_DOWN, LL_TIM_COUNTERMODE_CENTER_UP, LL_TIM_COUNTERMODE_CENTER_DOWN, LL_TIM_COUNTERMODE_CENTER_UP_DOWN }
资源使用	
说明	

2.25.13 LL_TIM_EnableARRPreload

2.25.13.1 功能介绍

使能 TIM 自动重载预装载。

2.25.13.2 接口定义

函数接口	void LL_TIM_EnableARRPreload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.14 LL_TIM_DisableARRPreload

2.25.14.1 功能介绍

禁止使能 TIM 自动重载预装载。

2.25.14.2 接口定义

函数接口	void LL_TIM_DisableARRPreload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.15 LL_TIM_IsEnabledARRPreload

2.25.15.1 功能介绍

检查是否使能 TIM 自动重载预装载。

2.25.15.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledARRPreload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.16 LL_TIM_SetClockDivision

2.25.16.1 功能介绍

设置时钟分频。

2.25.16.2 接口定义

函数接口	void LL_TIM_SetClockDivision (TIM_TypeDef * TIMx, uint32_t ClockDivision)
输入	TIM_TypeDef * TIMx uint32_t ClockDivision: { LL_TIM_CLOCKDIVISION_DIV1, LL_TIM_CLOCKDIVISION_DIV2, LL_TIM_CLOCKDIVISION_DIV4 }
输出	无
返回值	无
资源使用	
说明	

2.25.17 LL_TIM_GetClockDivision

2.25.17.1 功能介绍

获取时钟分频。

2.25.17.2 接口定义

函数接口	uint32_t LL_TIM_GetClockDivision (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_CLOCKDIVISION_DIV1, LL_TIM_CLOCKDIVISION_DIV2, LL_TIM_CLOCKDIVISION_DIV4 }
资源使用	
说明	

2.25.18 LL_TIM_SetCounter

2.25.18.1 功能介绍

设置计数器计数器值。

2.25.18.2 接口定义

函数接口	void LL_TIM_SetCounter (TIM_TypeDef * TIMx, uint32_t Counter)
输入	TIM_TypeDef * TIMx uint32_t Counter: [0x0000, 0xFFFF]
输出	无
返回值	无
资源使用	
说明	

2.25.19 LL_TIM_GetCounter

2.25.19.1 功能介绍

获取计数器计数器值。

2.25.19.2 接口定义

函数接口	uint32_t LL_TIM_GetCounter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0x0000, 0xFFFF]
资源使用	
说明	

2.25.20 LL_TIM_GetDirection

2.25.20.1 功能介绍

获取计数器计数方向。

2.25.20.2 接口定义

函数接口	uint32_t LL_TIM_GetDirection (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_COUNTERDIRECTION_UP, LL_TIM_COUNTERDIRECTION_DOWN }
资源使用	
说明	

2.25.21 LL_TIM_SetPrescaler

2.25.21.1 功能介绍

设置 TIM 预分频值。

2.25.21.2 接口定义

函数接口	void LL_TIM_SetPrescaler (TIM_TypeDef * TIMx, uint32_t Prescaler)
------	---

输入	TIM_TypeDef * TIMx uint32_t Prescaler: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.22 LL_TIM_GetPrescaler

2.25.22.1 功能介绍

获取 TIM 预分频值。

2.25.22.2 接口定义

函数接口	uint32_t LL_TIM_GetPrescaler (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.23 LL_TIM_SetAutoReload

2.25.23.1 功能介绍

设置 TIM 自动重载值。

2.25.23.2 接口定义

函数接口	void LL_TIM_SetAutoReload (TIM_TypeDef * TIMx, uint32_t AutoReload)
输入	TIM_TypeDef * TIMx uint32_t AutoReload: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.24 LL_TIM_GetAutoReload

2.25.24.1 功能介绍

获取 TIM 自动重载值。

2.25.24.2 接口定义

函数接口	uint32_t LL_TIM_GetAutoReload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]

资源使用	
说明	

2.25.25 LL_TIM_SetRepetitionCounter

2.25.25.1 功能介绍

设置 TIM 重复计数器值。

2.25.25.2 接口定义

函数接口	void LL_TIM_SetRepetitionCounter (TIM_TypeDef * TIMx, uint32_t RepetitionCounter)
输入	TIM_TypeDef * TIMx uint32_t RepetitionCounter: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.26 LL_TIM_GetRepetitionCounter

2.25.26.1 功能介绍

获取 TIM 重复计数器值。

2.25.26.2 接口定义

函数接口	uint32_t LL_TIM_GetRepetitionCounter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.27 LL_TIM_EnableUIFRemap

2.25.27.1 功能介绍

使能 UIF 状态位重映射。

2.25.27.2 接口定义

函数接口	void LL_TIM_EnableUIFRemap (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.28 LL_TIM_DisableUIFRemap

2.25.28.1 功能介绍

禁止使能 UIF 状态位重映射。

2.25.28.2 接口定义

函数接口	void LL_TIM_DisableUIFRemap (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.29 LL_TIM_IsEnabledUIFRemap

2.25.29.1 功能介绍

检查是否使能 UIF 状态位重映射。

2.25.29.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledUIFRemap (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.30 LL_TIM_IsActiveFlagUIFCopy

2.25.30.1 功能介绍

检查 UIF 副本。

2.25.30.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlagUIFCopy (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.31 LL_TIM_CC_EnablePreload

2.25.31.1 功能介绍

使能捕获/比较预装载控制。

2.25.31.2 接口定义

函数接口	void LL_TIM_CC_EnablePreload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.32 LL_TIM_CC_DisablePreload

2.25.32.1 功能介绍

禁止使能捕获/比较预装载控制。

2.25.32.2 接口定义

函数接口	void LL_TIM_CC_DisablePreload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.33 LL_TIM_CC_IsEnabledPreload

2.25.33.1 功能介绍

检查是否使能捕获/比较预装载控制。

2.25.33.2 接口定义

函数接口	uint32_t LL_TIM_CC_IsEnabledPreload (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.34 LL_TIM_CC_SetUpdate

2.25.34.1 功能介绍

设置捕获/比较控制更新选择。

2.25.34.2 接口定义

函数接口	void LL_TIM_CC_SetUpdate (TIM_TypeDef * TIMx, uint32_t CCUpdateSource)
输入	TIM_TypeDef * TIMx uint32_t CCUpdateSource: { LL_TIM_CCUPDATESOURCE_COMG_ONLY, LL_TIM_CCUPDATESOURCE_COMG_AND_TRGI }
输出	无
返回值	无
资源使用	
说明	

2.25.35 LL_TIM_CC_GetUpdate

2.25.35.1 功能介绍

获取捕获/比较控制更新选择。

2.25.35.2 接口定义

函数接口	uint32_t LL_TIM_CC_GetUpdate (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_CCUPDATESOURCE_COMG_ONLY, LL_TIM_CCUPDATESOURCE_COMG_AND_TRGI }
资源使用	
说明	

2.25.36LL_TIM_CC_SetDMAReqTrigger

2.25.36.1 功能介绍

设置捕获/比较 DMA 选择。

2.25.36.2 接口定义

函数接口	void LL_TIM_CC_SetDMAReqTrigger (TIM_TypeDef * TIMx, uint32_t DMAReqTrigger)
输入	TIM_TypeDef * TIMx uint32_t DMAReqTrigger: { LL_TIM_CCDMAREQUEST_CC, LL_TIM_CCDMAREQUEST_UPDATE }
输出	无
返回值	无
资源使用	
说明	

2.25.37LL_TIM_CC_GetDMAReqTrigger

2.25.37.1 功能介绍

获取捕获/比较 DMA 选择。

2.25.37.2 接口定义

函数接口	uint32_t LL_TIM_CC_GetDMAReqTrigger (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_CCDMAREQUEST_CC, LL_TIM_CCDMAREQUEST_UPDATE }
资源使用	
说明	

2.25.38LL_TIM_CC_SetLockLevel

2.25.38.1 功能介绍

设置锁定配置。

2.25.38.2 接口定义

函数接口	void LL_TIM_CC_SetLockLevel (TIM_TypeDef * TIMx, uint32_t LockLevel)
------	--

输入	TIM_TypeDef * TIMx uint32_t LockLevel: { LL_TIM_LOCKLEVEL_OFF, LL_TIM_LOCKLEVEL_1, LL_TIM_LOCKLEVEL_2, LL_TIM_LOCKLEVEL_3 }
输出	无
返回值	无
资源使用	
说明	

2.25.39 LL_TIM_CC_GetLockLevel

2.25.39.1 功能介绍

获取锁定配置。

2.25.39.2 接口定义

函数接口	uint32_t LL_TIM_CC_GetLockLevel (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_LOCKLEVEL_OFF, LL_TIM_LOCKLEVEL_1, LL_TIM_LOCKLEVEL_2, LL_TIM_LOCKLEVEL_3 }
资源使用	
说明	

2.25.40 LL_TIM_CC_EnableChannel

2.25.40.1 功能介绍

使能捕获/比较输出通道。

2.25.40.2 接口定义

函数接口	void LL_TIM_CC_EnableChannel (TIM_TypeDef * TIMx, uint32_t Channels)
输入	TIM_TypeDef * TIMx uint32_t Channels: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH1N, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH2N, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH3N, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.41 LL_TIM_CC_DisableChannel

2.25.41.1 功能介绍

禁止使能捕获/比较输出通道。

2.25.41.2 接口定义

函数接口	void LL_TIM_CC_DisableChannel (TIM_TypeDef * TIMx, uint32_t Channels)
输入	TIM_TypeDef * TIMx uint32_t Channels: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH1N, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH2N, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH3N, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.42 LL_TIM_CC_IsEnabledChannel
2.25.42.1 功能介绍

检查是否使能捕获/比较输出通道。

2.25.42.2 接口定义

函数接口	uint32_t LL_TIM_CC_IsEnabledChannel (TIM_TypeDef * TIMx, uint32_t Channels)
输入	TIM_TypeDef * TIMx uint32_t Channels: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH1N, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH2N, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH3N, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.43 LL_TIM_OC_ConfigOutput
2.25.43.1 功能介绍

设置输出比较配置。

2.25.43.2 接口定义

函数接口	void LL_TIM_OC_ConfigOutput (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Configuration)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4,

	LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 } uint32_t Configuration: { LL_TIM_OCPOLARITY_HIGH, LL_TIM_OCPOLARITY_LOW, LL_TIM_OCIDLESTATE_LOW, LL_TIM_OCIDLESTATE_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.25.44 LL_TIM_OC_SetMode

2.25.44.1 功能介绍

设置输出比较模式。

2.25.44.2 接口定义

函数接口	void LL_TIM_OC_SetMode (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Mode)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 } uint32_t Mode: { LL_TIM_OCMODE_FROZEN, LL_TIM_OCMODE_ACTIVE, LL_TIM_OCMODE_INACTIVE, LL_TIM_OCMODE_TOGGLE, LL_TIM_OCMODE_FORCED_INACTIVE, LL_TIM_OCMODE_FORCED_ACTIVE, LL_TIM_OCMODE_PWM1, LL_TIM_OCMODE_PWM2, LL_TIM_OCMODE_RETRIG_OPM1, LL_TIM_OCMODE_RETRIG_OPM2, LL_TIM_OCMODE_COMBINED_PWM1, LL_TIM_OCMODE_COMBINED_PWM2, LL_TIM_OCMODE_ASSYMETRIC_PWM1, LL_TIM_OCMODE_ASSYMETRIC_PWM2 }
输出	无
返回值	无
资源使用	
说明	

2.25.45 LL_TIM_OC_GetMode

2.25.45.1 功能介绍

获取输出比较模式。

2.25.45.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetMode (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel:

	{ LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	{ LL_TIM_OCMODE_FROZEN, LL_TIM_OCMODE_ACTIVE, LL_TIM_OCMODE_INACTIVE, LL_TIM_OCMODE_TOGGLE, LL_TIM_OCMODE_FORCED_INACTIVE, LL_TIM_OCMODE_FORCED_ACTIVE, LL_TIM_OCMODE_PWM1, LL_TIM_OCMODE_PWM2, LL_TIM_OCMODE_RETRIG_OPM1, LL_TIM_OCMODE_RETRIG_OPM2, LL_TIM_OCMODE_COMBINED_PWM1, LL_TIM_OCMODE_COMBINED_PWM2, LL_TIM_OCMODE_ASSYMETRIC_PWM1, LL_TIM_OCMODE_ASSYMETRIC_PWM2 }
资源使用	
说明	

2.25.46 LL_TIM_OC_SetPolarity

2.25.46.1 功能介绍

设置输出比较极性。

2.25.46.2 接口定义

函数接口	void LL_TIM_OC_SetPolarity (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Polarity)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 } uint32_t Polarity: { LL_TIM_OCPOLARITY_HIGH, LL_TIM_OCPOLARITY_LOW }
输出	无
返回值	无
资源使用	
说明	

2.25.47 LL_TIM_OC_GetPolarity

2.25.47.1 功能介绍

获取输出比较极性。

2.25.47.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetPolarity (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2,

	LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	{ LL_TIM_OCPOLARITY_HIGH, LL_TIM_OCPOLARITY_LOW }
资源使用	
说明	

2.25.48 LL_TIM_OC_SetIdleState

2.25.48.1 功能介绍

设置输出空闲状态。

2.25.48.2 接口定义

函数接口	void LL_TIM_OC_SetIdleState (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t IdleState)
输入	TIM_TypeDef * TIMx { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH1N, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH2N, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH3N, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 } uint32_t IdleState: { LL_TIM_OCPOLARITY_HIGH, LL_TIM_OCPOLARITY_LOW }
输出	无
返回值	无
资源使用	
说明	

2.25.49 LL_TIM_OC_GetIdleState

2.25.49.1 功能介绍

获取输出空闲状态。

2.25.49.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetIdleState (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channels: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH1N, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH2N, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH3N, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	{ LL_TIM_OCPOLARITY_HIGH, LL_TIM_OCPOLARITY_LOW }
资源使用	
说明	

2.25.50 LL_TIM_OC_EnableFast

2.25.50.1 功能介绍

使能输出通道的快速模式。

2.25.50.2 接口定义

函数接口	void LL_TIM_OC_EnableFast (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.51 LL_TIM_OC_DisableFast

2.25.51.1 功能介绍

禁止使能输出通道的快速模式。

2.25.51.2 接口定义

函数接口	void LL_TIM_OC_DisableFast (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.52 LL_TIM_OC_IsEnabledFast

2.25.52.1 功能介绍

检查是否使能输出通道的快速模式。

2.25.52.2 接口定义

函数接口	uint32_t LL_TIM_OC_IsEnabledFast (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }

输出	无
返回值	{0,1}
资源使用	
说明	

2.25.53 LL_TIM_OC_EnablePreload

2.25.53.1 功能介绍

使能比较寄存器输出通道预加载。

2.25.53.2 接口定义

函数接口	void LL_TIM_OC_EnablePreload (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.54 LL_TIM_OC_DisablePreload

2.25.54.1 功能介绍

禁止使能比较寄存器输出通道预加载。

2.25.54.2 接口定义

函数接口	void LL_TIM_OC_DisablePreload (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.55 LL_TIM_OC_IsEnabledPreload

2.25.55.1 功能介绍

检查是否使能比较寄存器输出通道预加载。

2.25.55.2 接口定义

函数接口	uint32_t LL_TIM_OC_IsEnabledPreload (TIM_TypeDef * TIMx, uint32_t Channel)
------	--

输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.56 LL_TIM_OC_EnableClear

2.25.56.1 功能介绍

使能清除外部事件的输出通道。

2.25.56.2 接口定义

函数接口	void LL_TIM_OC_EnableClear (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.57 LL_TIM_OC_DisableClear

2.25.57.1 功能介绍

禁止使能清除外部事件的输出通道。

2.25.57.2 接口定义

函数接口	void LL_TIM_OC_DisableClear (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	无
资源使用	
说明	

2.25.58 LL_TIM_OC_IsEnabledClear

2.25.58.1 功能介绍

检查是否使能清除外部事件的输出通道。

2.25.58.2 接口定义

函数接口	uint32_t LL_TIM_OC_IsEnabledClear (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.59 LL_TIM_OC_SetDeadTime

2.25.59.1 功能介绍

配置死区发生器时间。

2.25.59.2 接口定义

函数接口	void LL_TIM_OC_SetDeadTime (TIM_TypeDef * TIMx, uint32_t DeadTime)
输入	TIM_TypeDef * TIMx uint32_t DeadTime: [0, 255]
输出	无
返回值	无
资源使用	
说明	

2.25.60 LL_TIM_OC_GetDeadTime

2.25.60.1 功能介绍

获取死区发生器时间。

2.25.60.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetDeadTime (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 255]
资源使用	
说明	

2.25.61 LL_TIM_OC_SetCompareCH1

2.25.61.1 功能介绍

设置输出通道 1 的比较值。

2.25.61.2 接口定义

函数接口	void LL_TIM_OC_SetCompareCH1 (TIM_TypeDef * TIMx, uint32_t CompareValue)
输入	TIM_TypeDef * TIMx uint32_t CompareValue: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.62 LL_TIM_OC_SetCompareCH2

2.25.62.1 功能介绍

设置输出通道 2 的比较值。

2.25.62.2 接口定义

函数接口	void LL_TIM_OC_SetCompareCH2 (TIM_TypeDef * TIMx, uint32_t CompareValue)
输入	TIM_TypeDef * TIMx uint32_t CompareValue: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.63 LL_TIM_OC_SetCompareCH3

2.25.63.1 功能介绍

设置输出通道 3 的比较值。

2.25.63.2 接口定义

函数接口	void LL_TIM_OC_SetCompareCH3 (TIM_TypeDef * TIMx, uint32_t CompareValue)
输入	TIM_TypeDef * TIMx uint32_t CompareValue: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.64 LL_TIM_OC_SetCompareCH4

2.25.64.1 功能介绍

设置输出通道 4 的比较值。

2.25.64.2 接口定义

函数接口	void LL_TIM_OC_SetCompareCH4(TIM_TypeDef * TIMx, uint32_t CompareValue)
输入	TIM_TypeDef * TIMx uint32_t CompareValue: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.65 LL_TIM_OC_SetCompareCH5

2.25.65.1 功能介绍

设置输出通道 5 的比较值。

2.25.65.2 接口定义

函数接口	void LL_TIM_OC_SetCompareCH5 (TIM_TypeDef * TIMx, uint32_t CompareValue)
输入	TIM_TypeDef * TIMx uint32_t CompareValue: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.66 LL_TIM_OC_SetCompareCH6

2.25.66.1 功能介绍

设置输出通道 6 的比较值。

2.25.66.2 接口定义

函数接口	void LL_TIM_OC_SetCompareCH6 (TIM_TypeDef * TIMx, uint32_t CompareValue)
输入	TIM_TypeDef * TIMx uint32_t CompareValue: [0, 65535]
输出	无
返回值	无
资源使用	
说明	

2.25.67 LL_TIM_OC_GetCompareCH1

2.25.67.1 功能介绍

获取输出通道 1 的比较值。

2.25.67.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetCompareCH1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.68 LL_TIM_OC_GetCompareCH2

2.25.68.1 功能介绍

获取输出通道 2 的比较值。

2.25.68.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetCompareCH2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.69 LL_TIM_OC_GetCompareCH3

2.25.69.1 功能介绍

获取输出通道 3 的比较值。

2.25.69.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetCompareCH3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.70 LL_TIM_OC_GetCompareCH4

2.25.70.1 功能介绍

获取输出通道 4 的比较值。

2.25.70.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetCompareCH4(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	[0, 65535]
资源使用	
说明	

2.25.71 LL_TIM_OC_GetCompareCH5

2.25.71.1 功能介绍

获取输出通道 5 的比较值。

2.25.71.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetCompareCH5(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.72 LL_TIM_OC_GetCompareCH6

2.25.72.1 功能介绍

获取输出通道 6 的比较值。

2.25.72.2 接口定义

函数接口	uint32_t LL_TIM_OC_GetCompareCH6 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.73 LL_TIM_SetCH5CombinedChannels

2.25.73.1 功能介绍

选择通道 5 和哪个通道组合。

2.25.73.2 接口定义

函数接口	void LL_TIM_SetCH5CombinedChannels (TIM_TypeDef * TIMx, uint32_t GroupCH5)
输入	TIM_TypeDef * TIMx uint32_t GroupCH5: { LL_TIM_GROUPCH5_NONE, LL_TIM_GROUPCH5_OC1REFC, LL_TIM_GROUPCH5_OC2REFC, LL_TIM_GROUPCH5_OC3REFC }
输出	无
返回值	无
资源使用	
说明	

2.25.74 LL_TIM_GetCH5CombinedChannels

2.25.74.1 功能介绍

获取通道 5 和哪个通道组合。

2.25.74.2 接口定义

函数接口	uint32_t LL_TIM_GetCH5CombinedChannels (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_GROUPCH5_NONE, LL_TIM_GROUPCH5_OC1REFC, LL_TIM_GROUPCH5_OC2REFC, LL_TIM_GROUPCH5_OC3REFC }
资源使用	
说明	

2.25.75 LL_TIM_IC_Config

2.25.75.1 功能介绍

配置输入通道。

2.25.75.2 接口定义

函数接口	void LL_TIM_IC_Config (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ActiveInput, uint32_t InputPSC, uint32_t InputFilter, uint32_t InputPol)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 } uint32_t ActiveInput: {LL_TIM_ACTIVEINPUT_DIRECTTI, LL_TIM_ACTIVEINPUT_INDIRECTTI, LL_TIM_ACTIVEINPUT_TRC } uint32_t InputPSC: {LL_TIM_ICPSC_DIV1, LL_TIM_ICPSC_DIV2, LL_TIM_ICPSC_DIV4, LL_TIM_ICPSC_DIV8 } uint32_t InputFilter: {LL_TIM_IC_FILTER_FDIV1, LL_TIM_IC_FILTER_FDIV1_N2, LL_TIM_IC_FILTER_FDIV1_N4, LL_TIM_IC_FILTER_FDIV1_N8, LL_TIM_IC_FILTER_FDIV2_N6, LL_TIM_IC_FILTER_FDIV2_N8, LL_TIM_IC_FILTER_FDIV4_N6, LL_TIM_IC_FILTER_FDIV4_N8, LL_TIM_IC_FILTER_FDIV8_N6, LL_TIM_IC_FILTER_FDIV8_N8, LL_TIM_IC_FILTER_FDIV16_N5, LL_TIM_IC_FILTER_FDIV16_N6, LL_TIM_IC_FILTER_FDIV16_N8, LL_TIM_IC_FILTER_FDIV32_N5, LL_TIM_IC_FILTER_FDIV32_N6, LL_TIM_IC_FILTER_FDIV32_N8 } uint32_t InputPol: {LL_TIM_IC_POLARITY_RISING, LL_TIM_IC_POLARITY_FALLING, LL_TIM_IC_POLARITY_BOTHEDGE }

输出	无
返回值	无
资源使用	
说明	

2.25.76 LL_TIM_IC_SetActiveInput

2.25.76.1 功能介绍

设置通道使用的输入。

2.25.76.2 接口定义

函数接口	void LL_TIM_IC_SetActiveInput (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICActiveInput)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 } uint32_t ICActiveInput: {LL_TIM_ACTIVEINPUT_DIRECTTI, LL_TIM_ACTIVEINPUT_INDIRECTTI, LL_TIM_ACTIVEINPUT_TRC }
输出	无
返回值	无
资源使用	
说明	

2.25.77 LL_TIM_IC_GetActiveInput

2.25.77.1 功能介绍

获取通道使用的输入。

2.25.77.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetActiveInput (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 }
输出	无
返回值	{LL_TIM_ACTIVEINPUT_DIRECTTI, LL_TIM_ACTIVEINPUT_INDIRECTTI, LL_TIM_ACTIVEINPUT_TRC }
资源使用	
说明	

2.25.78 LL_TIM_IC_SetPrescaler

2.25.78.1 功能介绍

设置输入通道的预分频值。

2.25.78.2 接口定义

函数接口	void LL_TIM_IC_SetPrescaler (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICPrescaler)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 } uint32_t ICPrescaler: {LL_TIM_ICPSC_DIV1, LL_TIM_ICPSC_DIV2, LL_TIM_ICPSC_DIV4, LL_TIM_ICPSC_DIV8 }
输出	无
返回值	无
资源使用	
说明	

2.25.79LL_TIM_IC_GetPrescaler

2.25.79.1 功能介绍

获取输入通道的预分频值。

2.25.79.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetPrescaler (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 }
输出	无
返回值	{LL_TIM_ICPSC_DIV1, LL_TIM_ICPSC_DIV2, LL_TIM_ICPSC_DIV4, LL_TIM_ICPSC_DIV8 }
资源使用	
说明	

2.25.80LL_TIM_IC_SetFilter

2.25.80.1 功能介绍

设置输入滤波器持续时间。

2.25.80.2 接口定义

函数接口	void LL_TIM_IC_SetFilter (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICFilter)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 } uint32_t ICFilter: {LL_TIM_IC_FILTER_FDIV1, LL_TIM_IC_FILTER_FDIV1_N2,

	LL_TIM_IC_FILTER_FDIV1_N4, LL_TIM_IC_FILTER_FDIV1_N8, LL_TIM_IC_FILTER_FDIV2_N6, LL_TIM_IC_FILTER_FDIV2_N8, LL_TIM_IC_FILTER_FDIV4_N6, LL_TIM_IC_FILTER_FDIV4_N8, LL_TIM_IC_FILTER_FDIV8_N6, LL_TIM_IC_FILTER_FDIV8_N8, LL_TIM_IC_FILTER_FDIV16_N5, LL_TIM_IC_FILTER_FDIV16_N6, LL_TIM_IC_FILTER_FDIV16_N8, LL_TIM_IC_FILTER_FDIV32_N5, LL_TIM_IC_FILTER_FDIV32_N6, LL_TIM_IC_FILTER_FDIV32_N8 }
输出	无
返回值	无
资源使用	
说明	

2.25.81 LL_TIM_IC_GetFilter

2.25.81.1 功能介绍

获取输入滤波器持续时间。

2.25.81.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetFilter (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 }
输出	无
返回值	{LL_TIM_IC_FILTER_FDIV1, LL_TIM_IC_FILTER_FDIV1_N2, LL_TIM_IC_FILTER_FDIV1_N4, LL_TIM_IC_FILTER_FDIV1_N8, LL_TIM_IC_FILTER_FDIV2_N6, LL_TIM_IC_FILTER_FDIV2_N8, LL_TIM_IC_FILTER_FDIV4_N6, LL_TIM_IC_FILTER_FDIV4_N8, LL_TIM_IC_FILTER_FDIV8_N6, LL_TIM_IC_FILTER_FDIV8_N8, LL_TIM_IC_FILTER_FDIV16_N5, LL_TIM_IC_FILTER_FDIV16_N6, LL_TIM_IC_FILTER_FDIV16_N8, LL_TIM_IC_FILTER_FDIV32_N5, LL_TIM_IC_FILTER_FDIV32_N6, LL_TIM_IC_FILTER_FDIV32_N8 }
资源使用	
说明	

2.25.82 LL_TIM_IC_SetPolarity

2.25.82.1 功能介绍

设置输入通道极性。

2.25.82.2 接口定义

函数接口	void LL_TIM_IC_SetPolarity (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t ICPolarity)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 }

	uint32_t ICPolarity: { LL_TIM_IC_POLARITY_RISING, LL_TIM_IC_POLARITY_FALLING, LL_TIM_IC_POLARITY_BOTHEDGE }
输出	无
返回值	无
资源使用	
说明	

2.25.83 LL_TIM_IC_GetPolarity

2.25.83.1 功能介绍

获取输入通道极性。

2.25.83.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetPolarity (TIM_TypeDef * TIMx, uint32_t Channel)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 }
输出	无
返回值	{ LL_TIM_IC_POLARITY_RISING, LL_TIM_IC_POLARITY_FALLING, LL_TIM_IC_POLARITY_BOTHEDGE }
资源使用	
说明	

2.25.84 LL_TIM_IC_EnableXORCombination

2.25.84.1 功能介绍

TIM1_CH1、CH2 和 CH3 引脚连接到 TI1 输入。

2.25.84.2 接口定义

函数接口	void LL_TIM_IC_EnableXORCombination (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.85 LL_TIM_IC_DisableXORCombination

2.25.85.1 功能介绍

TIM1_CH1、CH2 和 CH3 引脚不连接到 TI1 输入。

2.25.85.2 接口定义

函数接口	void LL_TIM_IC_DisableXORCombination (TIM_TypeDef * TIMx)
------	---

输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.86LL_TIM_IC_IsEnabledXORCombination

2.25.86.1 功能介绍

检查 TIM1_CH1、CH2 和 CH3 引脚是否连接到 TI1 输入。

2.25.86.2 接口定义

函数接口	uint32_t LL_TIM_IC_IsEnabledXORCombination (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.87LL_TIM_IC_GetCaptureCH1

2.25.87.1 功能介绍

获取输入通道 1 的捕获值。

2.25.87.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetCaptureCH1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.88LL_TIM_IC_GetCaptureCH2

2.25.88.1 功能介绍

获取输入通道 2 的捕获值。

2.25.88.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetCaptureCH2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.89LL_TIM_IC_GetCaptureCH3

2.25.89.1 功能介绍

获取输入通道 3 的捕获值。

2.25.89.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetCaptureCH3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.90LL_TIM_IC_GetCaptureCH4

2.25.90.1 功能介绍

获取输入通道 4 的捕获值。

2.25.90.2 接口定义

函数接口	uint32_t LL_TIM_IC_GetCaptureCH4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	[0, 65535]
资源使用	
说明	

2.25.91LL_TIM_EnableExternalClock

2.25.91.1 功能介绍

使能外部时钟。

2.25.91.2 接口定义

函数接口	void LL_TIM_EnableExternalClock (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.92LL_TIM_DisableExternalClock

2.25.92.1 功能介绍

禁止使能外部时钟。

2.25.92.2 接口定义

函数接口	void LL_TIM_DisableExternalClock (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	无
资源使用	
说明	

2.25.93 LL_TIM_IsEnabledExternalClock

2.25.93.1 功能介绍

检查是否使能外部时钟。

2.25.93.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledExternalClock (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.94 LL_TIM_SetClockSource

2.25.94.1 功能介绍

设置计数器时钟的时钟源。

2.25.94.2 接口定义

函数接口	void LL_TIM_SetClockSource (TIM_TypeDef * TIMx, uint32_t ClockSource)
输入	TIM_TypeDef * TIMx uint32_t ClockSource: { LL_TIM_CLOCKSOURCE_INTERNAL, LL_TIM_CLOCKSOURCE_EXT_MODE1, LL_TIM_CLOCKSOURCE_EXT_MODE2 }
输出	无
返回值	无
资源使用	
说明	

2.25.95 LL_TIM_GetClockSource

2.25.95.1 功能介绍

获取计数器时钟的时钟源。

2.25.95.2 接口定义

函数接口	uint32_t LL_TIM_GetClockSource (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_CLOCKSOURCE_INTERNAL, LL_TIM_CLOCKSOURCE_EXT_MODE1, LL_TIM_CLOCKSOURCE_EXT_MODE2 }
资源使用	

说明	
----	--

2.25.96LL_TIM_SetEncoderMode

2.25.96.1 功能介绍

设置编码器接口模式。

2.25.96.2 接口定义

函数接口	void LL_TIM_SetEncoderMode (TIM_TypeDef * TIMx, uint32_t EncoderMode)
输入	TIM_TypeDef * TIMx uint32_t EncoderMode : { LL_TIM_ENCODERMODE_X2_TI1, LL_TIM_ENCODERMODE_X2_TI2, LL_TIM_ENCODERMODE_X4_TI12 }
输出	无
返回值	无
资源使用	
说明	

2.25.97LL_TIM_GetEncoderMode

2.25.97.1 功能介绍

获取编码器接口模式。

2.25.97.2 接口定义

函数接口	uint32_t LL_TIM_GetEncoderMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_ENCODERMODE_X2_TI1, LL_TIM_ENCODERMODE_X2_TI2, LL_TIM_ENCODERMODE_X4_TI12 }
资源使用	
说明	

2.25.98LL_TIM_SetTriggerOutput

2.25.98.1 功能介绍

设置用于定时器同步的触发器输出。

2.25.98.2 接口定义

函数接口	void LL_TIM_SetTriggerOutput (TIM_TypeDef * TIMx, uint32_t TimerSynchronization)
输入	TIM_TypeDef * TIMx uint32_t TimerSynchronization: { LL_TIM_TRGO_RESET, LL_TIM_TRGO_ENABLE, LL_TIM_TRGO_UPDATE, LL_TIM_TRGO_CC1IF, LL_TIM_TRGO_OC1REF, LL_TIM_TRGO_OC2REF, LL_TIM_TRGO_OC3REF, LL_TIM_TRGO_OC4REF }

输出	无
返回值	无
资源使用	
说明	

2.25.99 LL_TIM_GetTriggerOutput

2.25.99.1 功能介绍

获取用于定时器同步的触发器输出。

2.25.99.2 接口定义

函数接口	uint32_t LL_TIM_GetTriggerOutput (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_TRGO_RESET, LL_TIM_TRGO_ENABLE, LL_TIM_TRGO_UPDATE, LL_TIM_TRGO_CC1IF, LL_TIM_TRGO_OC1REF, LL_TIM_TRGO_OC2REF, LL_TIM_TRGO_OC3REF, LL_TIM_TRGO_OC4REF }
资源使用	
说明	

2.25.100 LL_TIM_SetTriggerOutput2

2.25.100.1 功能介绍

设置用于定时器同步的触发器输出 2。

2.25.100.2 接口定义

函数接口	void LL_TIM_SetTriggerOutput2 (TIM_TypeDef * TIMx, uint32_t ADCSynchronization)
输入	TIM_TypeDef * TIMx uint32_t ADCSynchronization: { LL_TIM_TRGO2_RESET, LL_TIM_TRGO2_ENABLE, LL_TIM_TRGO2_UPDATE, LL_TIM_TRGO2_CC1F, LL_TIM_TRGO2_OC1, LL_TIM_TRGO2_OC2, LL_TIM_TRGO2_OC3, LL_TIM_TRGO2_OC4, LL_TIM_TRGO2_OC5, LL_TIM_TRGO2_OC6, LL_TIM_TRGO2_OC4_RISINGFALLING, LL_TIM_TRGO2_OC6_RISINGFALLING, LL_TIM_TRGO2_OC4_RISING_OC6_RISING, LL_TIM_TRGO2_OC4_RISING_OC6_FALLING, LL_TIM_TRGO2_OC5_RISING_OC6_RISING, LL_TIM_TRGO2_OC5_RISING_OC6_FALLING }
输出	无
返回值	无
资源使用	
说明	

2.25.101 LL_TIM_GetTriggerOutput2

2.25.101.1 功能介绍

获取用于定时器同步的触发器输出 2。

2.25.101.2 接口定义

函数接口	uint32_t LL_TIM_GetTriggerOutput2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_TRGO2_RESET, LL_TIM_TRGO2_ENABLE, LL_TIM_TRGO2_UPDATE, LL_TIM_TRGO2_CC1F, LL_TIM_TRGO2_OC1, LL_TIM_TRGO2_OC2, LL_TIM_TRGO2_OC3, LL_TIM_TRGO2_OC4, LL_TIM_TRGO2_OC5, LL_TIM_TRGO2_OC6, LL_TIM_TRGO2_OC4_RISINGFALLING, LL_TIM_TRGO2_OC6_RISINGFALLING, LL_TIM_TRGO2_OC4_RISING_OC6_RISING, LL_TIM_TRGO2_OC4_RISING_OC6_FALLING, LL_TIM_TRGO2_OC5_RISING_OC6_RISING, LL_TIM_TRGO2_OC5_RISING_OC6_FALLING }
资源使用	
说明	

2.25.102 LL_TIM_SetSlaveMode

2.25.102.1 功能介绍

设置从模式选择。

2.25.102.2 接口定义

函数接口	void LL_TIM_SetSlaveMode (TIM_TypeDef * TIMx, uint32_t SlaveMode)
输入	TIM_TypeDef * TIMx uint32_t SlaveMode: { LL_TIM_SLAVEMODE_DISABLED, LL_TIM_SLAVEMODE_RESET, LL_TIM_SLAVEMODE_GATED, LL_TIM_SLAVEMODE_TRIGGER, LL_TIM_SLAVEMODE_EXTERNAL1, LL_TIM_SLAVEMODE_COMBINED_RESETTRIGGER }
输出	无
返回值	无
资源使用	
说明	

2.25.103 LL_TIM_GetSlaveMode

2.25.103.1 功能介绍

获取从模式选择。

2.25.103.2 接口定义

函数接口	uint32_t LL_TIM_GetSlaveMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_SLAVEMODE_DISABLED, LL_TIM_SLAVEMODE_RESET, LL_TIM_SLAVEMODE_GATED, LL_TIM_SLAVEMODE_TRIGGER, LL_TIM_SLAVEMODE_EXTERNAL1, LL_TIM_SLAVEMODE_COMBINED_RESETTRIGGER }
资源使用	
说明	

2.25.104 LL_TIM_SetTriggerInput

2.25.104.1 功能介绍

设置选择用于同步计数器的触发器输入。

2.25.104.2 接口定义

函数接口	void LL_TIM_SetTriggerInput (TIM_TypeDef * TIMx, uint32_t TriggerInput)
输入	TIM_TypeDef * TIMx uint32_t TriggerInput: { LL_TIM_TS_ITR0, LL_TIM_TS_ITR1, LL_TIM_TS_ITR2, LL_TIM_TS_ITR3, LL_TIM_TS_TI1F_ED, LL_TIM_TS_TI1FP1, LL_TIM_TS_TI2FP2, LL_TIM_TS_ETRF }
输出	无
返回值	无
资源使用	
说明	

2.25.105 LL_TIM_GetTriggerInput

2.25.105.1 功能介绍

获取选择用于同步计数器的触发器输入。

2.25.105.2 接口定义

函数接口	void LL_TIM_SetTriggerInput (TIM_TypeDef * TIMx, uint32_t TriggerInput)
输入	TIM_TypeDef * TIMx uint32_t TriggerInput: { LL_TIM_TS_ITR0, LL_TIM_TS_ITR1, LL_TIM_TS_ITR2, LL_TIM_TS_ITR3, LL_TIM_TS_TI1F_ED, LL_TIM_TS_TI1FP1, LL_TIM_TS_TI2FP2, LL_TIM_TS_ETRF }
输出	无
返回值	无
资源使用	

说明	
----	--

2.25.106 LL_TIM_EnableMasterSlaveMode

2.25.106.1 功能介绍

使能主/从模式。

2.25.106.2 接口定义

函数接口	void LL_TIM_EnableMasterSlaveMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.107 LL_TIM_DisableMasterSlaveMode

2.25.107.1 功能介绍

禁止使能主/从模式。

2.25.107.2 接口定义

函数接口	void LL_TIM_DisableMasterSlaveMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.108 LL_TIM_IsEnabledMasterSlaveMode

2.25.108.1 功能介绍

检查是否使能主/从模式。

2.25.108.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledMasterSlaveMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.109 LL_TIM_ConfigETR

2.25.109.1 功能介绍

配置外部触发输入。

2.25.109.2 接口定义

函数接口	void LL_TIM_ConfigETR (TIM_TypeDef * TIMx, uint32_t ETRPolarity, uint32_t ETRPrescaler, uint32_t ETRFilter)
------	---

输入	<pre>TIM_TypeDef * TIMx uint32_t ETRPolarity: {LL_TIM_ETR_POLARITY_NONINVERTED, LL_TIM_ETR_POLARITY_INVERTED } uint32_t ETRPrescaler: { LL_TIM_ETR_PRESCALER_DIV1, LL_TIM_ETR_PRESCALER_DIV2, LL_TIM_ETR_PRESCALER_DIV4, LL_TIM_ETR_PRESCALER_DIV8 } uint32_t ETRFilter: {LL_TIM_ETR_FILTER_FDIV1, LL_TIM_ETR_FILTER_FDIV1_N2, LL_TIM_ETR_FILTER_FDIV1_N4, LL_TIM_ETR_FILTER_FDIV1_N8, LL_TIM_ETR_FILTER_FDIV2_N6, LL_TIM_ETR_FILTER_FDIV2_N8, LL_TIM_ETR_FILTER_FDIV4_N6, LL_TIM_ETR_FILTER_FDIV4_N8, LL_TIM_ETR_FILTER_FDIV8_N6, LL_TIM_ETR_FILTER_FDIV8_N8, LL_TIM_ETR_FILTER_FDIV16_N5, LL_TIM_ETR_FILTER_FDIV16_N6, LL_TIM_ETR_FILTER_FDIV16_N8, LL_TIM_ETR_FILTER_FDIV32_N5, LL_TIM_ETR_FILTER_FDIV32_N6, LL_TIM_ETR_FILTER_FDIV32_N8 }</pre>
输出	无
返回值	无
资源使用	
说明	

2.25.110 LL_TIM_GetETRFilter

2.25.110.1 功能介绍

获取外部触发输入滤波。

2.25.110.2 接口定义

函数接口	uint32_t LL_TIM_GetETRFilter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	<pre>{LL_TIM_ETR_FILTER_FDIV1, LL_TIM_ETR_FILTER_FDIV1_N2, LL_TIM_ETR_FILTER_FDIV1_N4, LL_TIM_ETR_FILTER_FDIV1_N8, LL_TIM_ETR_FILTER_FDIV2_N6, LL_TIM_ETR_FILTER_FDIV2_N8, LL_TIM_ETR_FILTER_FDIV4_N6, LL_TIM_ETR_FILTER_FDIV4_N8, LL_TIM_ETR_FILTER_FDIV8_N6, LL_TIM_ETR_FILTER_FDIV8_N8, LL_TIM_ETR_FILTER_FDIV16_N5, LL_TIM_ETR_FILTER_FDIV16_N6, LL_TIM_ETR_FILTER_FDIV16_N8, LL_TIM_ETR_FILTER_FDIV32_N5, LL_TIM_ETR_FILTER_FDIV32_N6, LL_TIM_ETR_FILTER_FDIV32_N8 }</pre>

资源使用	
说明	

2.25.111 LL_TIM_GetETRPrescale

2.25.111.1 功能介绍

获取外部触发输入分频值。

2.25.111.2 接口定义

函数接口	uint32_t LL_TIM_GetETRPrescale (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_ETR_PRESCALER_DIV1, LL_TIM_ETR_PRESCALER_DIV2, LL_TIM_ETR_PRESCALER_DIV4, LL_TIM_ETR_PRESCALER_DIV8 }
资源使用	
说明	

2.25.112 LL_TIM_GetETRPolarity

2.25.112.1 功能介绍

获取外部触发输入极性。

2.25.112.2 接口定义

函数接口	uint32_t LL_TIM_GetETRPolarity (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_ETR_POLARITY_NONINVERTED, LL_TIM_ETR_POLARITY_INVERTED }
资源使用	
说明	

2.25.113 LL_TIM_SetETRSource

2.25.113.1 功能介绍

设置外部触发输入源。

2.25.113.2 接口定义

函数接口	void LL_TIM_SetETRSource (TIM_TypeDef * TIMx, uint32_t ETRSource)
输入	TIM_TypeDef * TIMx uint32_t ETRSource: { TIM1: LL_TIM_ETRSOURCE_GPIO, LL_TIM_ETRSOURCE_COMP1, LL_TIM_ETRSOURCE_COMP2, LL_TIM_ETRSOURCE_ADC1_AWD1, LL_TIM_ETRSOURCE_COMP3, LL_TIM_ETRSOURCE_COMP4, TIM2 TIM3 TIM4:

	LL_TIM_ETRSOURCE_GPIO, LL_TIM_ETRSOURCE_COMP1, LL_TIM_ETRSOURCE_COMP2, LL_TIM_ETRSOURCE_LSE }
输出	无
返回值	无
资源使用	
说明	

2.25.114 LL_TIM_GetETRSource

2.25.114.1 功能介绍

获取外部触发输入源。

2.25.114.2 接口定义

函数接口	uint32_t LL_TIM_GetETRSource (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ TIM1: LL_TIM_ETRSOURCE_GPIO, LL_TIM_ETRSOURCE_COMP1, LL_TIM_ETRSOURCE_COMP2, LL_TIM_ETRSOURCE_ADC1_AWD1, LL_TIM_ETRSOURCE_COMP3, LL_TIM_ETRSOURCE_COMP4, TIM2 TIM3 TIM4: LL_TIM_ETRSOURCE_GPIO, LL_TIM_ETRSOURCE_COMP1, LL_TIM_ETRSOURCE_COMP2, LL_TIM_ETRSOURCE_LSE }
资源使用	
说明	

2.25.115 LL_TIM_EnableBRK

2.25.115.1 功能介绍

使能断路。

2.25.115.2 接口定义

函数接口	void LL_TIM_EnableBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.116 LL_TIM_DisableBRK

2.25.116.1 功能介绍

禁止使能断路。

2.25.116.2 接口定义

函数接口	void LL_TIM_DisableBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx

输出	无
返回值	无
资源使用	
说明	

2.25.117 LL_TIM_IsEnabledBRK

2.25.117.1 功能介绍

检查是否使能断路。

2.25.117.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.118 LL_TIM_ConfigBRK

2.25.118.1 功能介绍

配置断路输入。

2.25.118.2 接口定义

函数接口	void LL_TIM_ConfigBRK (TIM_TypeDef * TIMx, uint32_t BreakPolarity, uint32_t BreakFilter, uint32_t BreakAFMode)
输入	TIM_TypeDef * TIMx uint32_t BreakPolarity: { LL_TIM_BREAK_POLARITY_LOW, LL_TIM_BREAK_POLARITY_HIGH } uint32_t BreakFilter: { LL_TIM_BREAK_FILTER_FDIV1, LL_TIM_BREAK_FILTER_FDIV1_N2, LL_TIM_BREAK_FILTER_FDIV1_N4, LL_TIM_BREAK_FILTER_FDIV1_N8, LL_TIM_BREAK_FILTER_FDIV2_N6, LL_TIM_BREAK_FILTER_FDIV2_N8, LL_TIM_BREAK_FILTER_FDIV4_N6, LL_TIM_BREAK_FILTER_FDIV4_N8, LL_TIM_BREAK_FILTER_FDIV8_N6, LL_TIM_BREAK_FILTER_FDIV8_N8, LL_TIM_BREAK_FILTER_FDIV16_N5, LL_TIM_BREAK_FILTER_FDIV16_N6, LL_TIM_BREAK_FILTER_FDIV16_N8, LL_TIM_BREAK_FILTER_FDIV32_N5, LL_TIM_BREAK_FILTER_FDIV32_N6, LL_TIM_BREAK_FILTER_FDIV32_N8 }

	uint32_t BreakAFMode: { LL_TIM_BREAK_AFMODE_INPUT, LL_TIM_BREAK_AFMODE_BIDIRECTIONAL }
输出	无
返回值	无
资源使用	
说明	

2.25.119 LL_TIM_GetBRKPolarity

2.25.119.1 功能介绍

获取断路输入极性。

2.25.119.2 接口定义

函数接口	uint32_t LL_TIM_GetBRKPolarity (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_BREAK_POLARITY_LOW, LL_TIM_BREAK_POLARITY_HIGH }
资源使用	
说明	

2.25.120 LL_TIM_GetBRKAFMode

2.25.120.1 功能介绍

获取断路输入复用模式。

2.25.120.2 接口定义

函数接口	uint32_t LL_TIM_GetBRKAFMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_BREAK_AFMODE_INPUT, LL_TIM_BREAK_AFMODE_BIDIRECTIONAL }
资源使用	
说明	

2.25.121 LL_TIM_GetBRKFilter

2.25.121.1 功能介绍

获取断路输入滤波。

2.25.121.2 接口定义

函数接口	uint32_t LL_TIM_GetBRKFilter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_BREAK_FILTER_FDIV1, LL_TIM_BREAK_FILTER_FDIV1_N2, LL_TIM_BREAK_FILTER_FDIV1_N4,

	LL_TIM_BREAK_FILTER_FDIV1_N8, LL_TIM_BREAK_FILTER_FDIV2_N6, LL_TIM_BREAK_FILTER_FDIV2_N8, LL_TIM_BREAK_FILTER_FDIV4_N6, LL_TIM_BREAK_FILTER_FDIV4_N8, LL_TIM_BREAK_FILTER_FDIV8_N6, LL_TIM_BREAK_FILTER_FDIV8_N8, LL_TIM_BREAK_FILTER_FDIV16_N5, LL_TIM_BREAK_FILTER_FDIV16_N6, LL_TIM_BREAK_FILTER_FDIV16_N8, LL_TIM_BREAK_FILTER_FDIV32_N5, LL_TIM_BREAK_FILTER_FDIV32_N6, LL_TIM_BREAK_FILTER_FDIV32_N8 }
资源使用	
说明	

2.25.122 LL_TIM_DisarmBRK

2.25.122.1 功能介绍

解除断路 2 输入。

2.25.122.2 接口定义

函数接口	void LL_TIM_DisarmBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.123 LL_TIM_ReArmBRK

2.25.123.1 功能介绍

启动断路 2 输入。

2.25.123.2 接口定义

函数接口	void LL_TIM_ReArmBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.124 LL_TIM_IsDisarmedBRK

2.25.124.1 功能介绍

检查是否解除断路 2 输入。

2.25.124.2 接口定义

函数接口	uint32_t LL_TIM_IsDisarmedBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.125 LL_TIM_EnableBRK2

2.25.125.1 功能介绍

使能断路 2。

2.25.125.2 接口定义

函数接口	void LL_TIM_EnableBRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.126 LL_TIM_DisableBRK2

2.25.126.1 功能介绍

禁止使能断路 2。

2.25.126.2 接口定义

函数接口	void LL_TIM_DisableBRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.127 LL_TIM_IsEnabledBRK2

2.25.127.1 功能介绍

检查是否使能断路 2。

2.25.127.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledBRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.128 LL_TIM_ConfigBRK2

2.25.128.1 功能介绍

配置断路 2 输入。

2.25.128.2 接口定义

函数接口	void LL_TIM_ConfigBRK2 (TIM_TypeDef * TIMx, uint32_t Break2Polarity, uint32_t Break2Filter, uint32_t Break2AFMode)
输入	TIM_TypeDef * TIMx uint32_t Break2Polarity: { LL_TIM_BREAK2_POLARITY_LOW, LL_TIM_BREAK2_POLARITY_HIGH } uint32_t Break2Filter: { LL_TIM_BREAK2_FILTER_FDIV1, LL_TIM_BREAK2_FILTER_FDIV1_N2, LL_TIM_BREAK2_FILTER_FDIV1_N4, LL_TIM_BREAK2_FILTER_FDIV1_N8, LL_TIM_BREAK2_FILTER_FDIV2_N6, LL_TIM_BREAK2_FILTER_FDIV2_N8, LL_TIM_BREAK2_FILTER_FDIV4_N6, LL_TIM_BREAK2_FILTER_FDIV4_N8, LL_TIM_BREAK2_FILTER_FDIV8_N6, LL_TIM_BREAK2_FILTER_FDIV8_N8, LL_TIM_BREAK2_FILTER_FDIV16_N5, LL_TIM_BREAK2_FILTER_FDIV16_N6, LL_TIM_BREAK2_FILTER_FDIV16_N8, LL_TIM_BREAK2_FILTER_FDIV32_N5, LL_TIM_BREAK2_FILTER_FDIV32_N6, LL_TIM_BREAK2_FILTER_FDIV32_N8 } uint32_t Break2AFMode: { LL_TIM_BREAK2_AFMODE_INPUT, LL_TIM_BREAK2_AFMODE_BIDIRECTIONAL }
输出	无
返回值	无
资源使用	
说明	

2.25.129 LL_TIM_GetBRK2Polarity

2.25.129.1 功能介绍

获取断路 2 输入极性。

2.25.129.2 接口定义

函数接口	uint32_t LL_TIM_GetBRK2Polarity (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_BREAK2_POLARITY_LOW,

	LL_TIM_BREAK2_POLARITY_HIGH }
资源使用	
说明	

2.25.130 LL_TIM_GetBRK2AFMode

2.25.130.1 功能介绍

获取断路 2 输入复用模式。

2.25.130.2 接口定义

函数接口	uint32_t LL_TIM_GetBRK2AFMode (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_BREAK2_AFMODE_INPUT, LL_TIM_BREAK2_AFMODE_BIDIRECTIONAL }
资源使用	
说明	

2.25.131 LL_TIM_GetBRK2Filter

2.25.131.1 功能介绍

获取断路 2 输入滤波。

2.25.131.2 接口定义

函数接口	uint32_t LL_TIM_GetBRK2Filter (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_BREAK2_FILTER_FDIV1, LL_TIM_BREAK2_FILTER_FDIV1_N2, LL_TIM_BREAK2_FILTER_FDIV1_N4, LL_TIM_BREAK2_FILTER_FDIV1_N8, LL_TIM_BREAK2_FILTER_FDIV2_N6, LL_TIM_BREAK2_FILTER_FDIV2_N8, LL_TIM_BREAK2_FILTER_FDIV4_N6, LL_TIM_BREAK2_FILTER_FDIV4_N8, LL_TIM_BREAK2_FILTER_FDIV8_N6, LL_TIM_BREAK2_FILTER_FDIV8_N8, LL_TIM_BREAK2_FILTER_FDIV16_N5, LL_TIM_BREAK2_FILTER_FDIV16_N6, LL_TIM_BREAK2_FILTER_FDIV16_N8, LL_TIM_BREAK2_FILTER_FDIV32_N5, LL_TIM_BREAK2_FILTER_FDIV32_N6, LL_TIM_BREAK2_FILTER_FDIV32_N8 }
资源使用	
说明	

2.25.132 LL_TIM_DisarmBRK2

2.25.132.1 功能介绍

解除断路 2 输入。

2.25.132.2 接口定义

函数接口	void LL_TIM_DisarmBRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.133 LL_TIM_ReArmBRK2

2.25.133.1 功能介绍

启动断路 2 输入。

2.25.133.2 接口定义

函数接口	void LL_TIM_ReArmBRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.134 LL_TIM_IsDisarmedBRK2

2.25.134.1 功能介绍

检查是否解除断路输入。

2.25.134.2 接口定义

函数接口	uint32_t LL_TIM_IsDisarmedBRK2(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.135 LL_TIM_SetOffStates

2.25.135.1 功能介绍

设置空闲模式下的关闭状态选择。

2.25.135.2 接口定义

函数接口	void LL_TIM_SetOffStates (TIM_TypeDef * TIMx, uint32_t OffStateIdle, uint32_t OffStateRun)
输入	TIM_TypeDef * TIMx uint32_t OffStateIdle:

	{ LL_TIM_OSSI_DISABLE, LL_TIM_OSSI_ENABLE } uint32_t OffStateRun: { LL_TIM_OSSR_DISABLE, LL_TIM_OSSR_ENABLE }
输出	无
返回值	无
资源使用	
说明	

2.25.136 LL_TIM_GetOffStates

2.25.136.1 功能介绍

获取空闲模式下的关闭状态选择。

2.25.136.2 接口定义

函数接口	uint32_t LL_TIM_GetOffStates (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_OSSI_DISABLE, LL_TIM_OSSI_ENABLE, LL_TIM_OSSR_DISABLE, LL_TIM_OSSR_ENABLE }
资源使用	
说明	

2.25.137 LL_TIM_EnableAutomaticOutput

2.25.137.1 功能介绍

使能自动输出。

2.25.137.2 接口定义

函数接口	void LL_TIM_EnableAutomaticOutput (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.138 LL_TIM_DisableAutomaticOutput

2.25.138.1 功能介绍

禁止使能自动输出。

2.25.138.2 接口定义

函数接口	void LL_TIM_DisableAutomaticOutput (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.139 LL_TIM_IsEnabledAutomaticOutput

2.25.139.1 功能介绍

检查是否使能自动输出。

2.25.139.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledAutomaticOutput (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.140 LL_TIM_EnableAllOutputs

2.25.140.1 功能介绍

使能主输出。

2.25.140.2 接口定义

函数接口	void LL_TIM_EnableAllOutputs (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.141 LL_TIM_DisableAllOutputs

2.25.141.1 功能介绍

禁止使能主输出。

2.25.141.2 接口定义

函数接口	void LL_TIM_DisableAllOutputs (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.142 LL_TIM_IsEnabledAllOutputs

2.25.142.1 功能介绍

检查是否使能主输出。

2.25.142.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledAllOutputs (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	{0,1}
资源使用	
说明	

2.25.143 LL_TIM_EnableBreakInputSource

2.25.143.1 功能介绍

使能信号连接到指定定时器中断输入。

2.25.143.2 接口定义

函数接口	void LL_TIM_EnableBreakInputSource (TIM_TypeDef * TIMx, uint32_t BreakInput, uint32_t Source)
输入	TIM_TypeDef * TIMx uint32_t BreakInput: { LL_TIM_BREAK_INPUT_BKIN, LL_TIM_BREAK_INPUT_BKIN2 } uint32_t Source: { LL_TIM_BKIN_SOURCE_BKIN, LL_TIM_BKIN_SOURCE_BKCOMP1, LL_TIM_BKIN_SOURCE_BKCOMP2 }
输出	无
返回值	无
资源使用	
说明	

2.25.144 LL_TIM_DisableBreakInputSource

2.25.144.1 功能介绍

禁止使能信号连接到指定定时器中断输入。

2.25.144.2 接口定义

函数接口	void LL_TIM_DisableBreakInputSource (TIM_TypeDef * TIMx, uint32_t BreakInput, uint32_t Source)
输入	TIM_TypeDef * TIMx uint32_t BreakInput: { LL_TIM_BREAK_INPUT_BKIN, LL_TIM_BREAK_INPUT_BKIN2 } uint32_t Source: { LL_TIM_BKIN_SOURCE_BKIN, LL_TIM_BKIN_SOURCE_BKCOMP1, LL_TIM_BKIN_SOURCE_BKCOMP2 }
输出	无
返回值	无
资源使用	
说明	

2.25.145 LL_TIM_IsEnabledBreakInputSource

2.25.145.1 功能介绍

检查是否使能信号连接到指定定时器中断输入。

2.25.145.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledBreakInputSource (TIM_TypeDef * TIMx, uint32_t BreakInput, uint32_t Source)
输入	TIM_TypeDef * TIMx uint32_t BreakInput: { LL_TIM_BREAK_INPUT_BKIN, LL_TIM_BREAK_INPUT_BKIN2 } uint32_t Source: { LL_TIM_BKIN_SOURCE_BKIN, LL_TIM_BKIN_SOURCE_BKCOMP1, LL_TIM_BKIN_SOURCE_BKCOMP2 }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.146 LL_TIM_SetBreakInputSourcePolarity

2.25.146.1 功能介绍

设置定时器断路输入断路信号的极性。

2.25.146.2 接口定义

函数接口	void LL_TIM_SetBreakInputSourcePolarity (TIM_TypeDef * TIMx, uint32_t BreakInput, uint32_t Source, uint32_t Polarity)
输入	TIM_TypeDef * TIMx uint32_t BreakInput: { LL_TIM_BREAK_INPUT_BKIN, LL_TIM_BREAK_INPUT_BKIN2 } uint32_t Source: { LL_TIM_BKIN_SOURCE_BKIN, LL_TIM_BKIN_SOURCE_BKCOMP1, LL_TIM_BKIN_SOURCE_BKCOMP2 } uint32_t Polarity: { LL_TIM_BKIN_POLARITY_LOW, LL_TIM_BKIN_POLARITY_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.25.147 LL_TIM_GetBreakInputSourcePolarity

2.25.147.1 功能介绍

获取定时器断路输入断路信号的极性。

2.25.147.2 接口定义

函数接口	uint32_t LL_TIM_GetBreakInputSourcePolarity (TIM_TypeDef * TIMx, uint32_t BreakInput, uint32_t Source)
输入	TIM_TypeDef * TIMx

	uint32_t BreakInput: { LL_TIM_BREAK_INPUT_BKIN, LL_TIM_BREAK_INPUT_BKIN2 } uint32_t Source: { LL_TIM_BKIN_SOURCE_BKIN, LL_TIM_BKIN_SOURCE_BKCOMP1, LL_TIM_BKIN_SOURCE_BKCOMP2 }
输出	无
返回值	{ LL_TIM_BKIN_POLARITY_LOW, LL_TIM_BKIN_POLARITY_HIGH }
资源使用	
说明	

2.25.148 LL_TIM_ConfigDMABurst

2.25.148.1 功能介绍

配置 DMA 基址和 DMA 连续传送长度。

2.25.148.2 接口定义

函数接口	void LL_TIM_ConfigDMABurst (TIM_TypeDef * TIMx, uint32_t DMABurstBaseAddress, uint32_t DMABurstLength)
输入	TIM_TypeDef * TIMx uint32_t DMABurstBaseAddress: { LL_TIM_DMABURST_BASEADDR_CR1, LL_TIM_DMABURST_BASEADDR_CR2, LL_TIM_DMABURST_BASEADDR_SMCR, LL_TIM_DMABURST_BASEADDR_DIER, LL_TIM_DMABURST_BASEADDR_SR, LL_TIM_DMABURST_BASEADDR_EGR, LL_TIM_DMABURST_BASEADDR_CCMR1, LL_TIM_DMABURST_BASEADDR_CCMR2, LL_TIM_DMABURST_BASEADDR_CCER, LL_TIM_DMABURST_BASEADDR_CNT, LL_TIM_DMABURST_BASEADDR_PSC, LL_TIM_DMABURST_BASEADDR_ARR, LL_TIM_DMABURST_BASEADDR_RCR, LL_TIM_DMABURST_BASEADDR_CCR1, LL_TIM_DMABURST_BASEADDR_CCR2, LL_TIM_DMABURST_BASEADDR_CCR3, LL_TIM_DMABURST_BASEADDR_CCR4, LL_TIM_DMABURST_BASEADDR_BDTR, LL_TIM_DMABURST_BASEADDR_OR1, LL_TIM_DMABURST_BASEADDR_CCMR3, LL_TIM_DMABURST_BASEADDR_CCR5, LL_TIM_DMABURST_BASEADDR_CCR6, LL_TIM_DMABURST_BASEADDR_AF1, LL_TIM_DMABURST_BASEADDR_AF2, LL_TIM_DMABURST_BASEADDR_TISEL }

	<pre>uint32_t DMABurstLength: { LL_TIM_DMABURST_LENGTH_1TRANSFER, LL_TIM_DMABURST_LENGTH_2TRANSFERS, LL_TIM_DMABURST_LENGTH_3TRANSFERS, LL_TIM_DMABURST_LENGTH_4TRANSFERS, LL_TIM_DMABURST_LENGTH_5TRANSFERS, LL_TIM_DMABURST_LENGTH_6TRANSFERS, LL_TIM_DMABURST_LENGTH_7TRANSFERS, LL_TIM_DMABURST_LENGTH_8TRANSFERS, LL_TIM_DMABURST_LENGTH_9TRANSFERS, LL_TIM_DMABURST_LENGTH_10TRANSFERS, LL_TIM_DMABURST_LENGTH_11TRANSFERS, LL_TIM_DMABURST_LENGTH_12TRANSFERS, LL_TIM_DMABURST_LENGTH_13TRANSFERS, LL_TIM_DMABURST_LENGTH_14TRANSFERS, LL_TIM_DMABURST_LENGTH_15TRANSFERS, LL_TIM_DMABURST_LENGTH_16TRANSFERS, LL_TIM_DMABURST_LENGTH_17TRANSFERS, LL_TIM_DMABURST_LENGTH_18TRANSFERS }</pre>
输出	无
返回值	无
资源使用	
说明	

2.25.149 LL_TIM_GetDMABurstLength

2.25.149.1 功能介绍

获取 DMA 连续传送长度。

2.25.149.2 接口定义

函数接口	uint32_t LL_TIM_GetDMABurstLength (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	<pre>{ LL_TIM_DMABURST_LENGTH_1TRANSFER, LL_TIM_DMABURST_LENGTH_2TRANSFERS, LL_TIM_DMABURST_LENGTH_3TRANSFERS, LL_TIM_DMABURST_LENGTH_4TRANSFERS, LL_TIM_DMABURST_LENGTH_5TRANSFERS, LL_TIM_DMABURST_LENGTH_6TRANSFERS, LL_TIM_DMABURST_LENGTH_7TRANSFERS, LL_TIM_DMABURST_LENGTH_8TRANSFERS, LL_TIM_DMABURST_LENGTH_9TRANSFERS, LL_TIM_DMABURST_LENGTH_10TRANSFERS, LL_TIM_DMABURST_LENGTH_11TRANSFERS, LL_TIM_DMABURST_LENGTH_12TRANSFERS, LL_TIM_DMABURST_LENGTH_13TRANSFERS,</pre>

	LL_TIM_DMABURST_LENGTH_14TRANSFERS, LL_TIM_DMABURST_LENGTH_15TRANSFERS, LL_TIM_DMABURST_LENGTH_16TRANSFERS, LL_TIM_DMABURST_LENGTH_17TRANSFERS, LL_TIM_DMABURST_LENGTH_18TRANSFERS }
资源使用	
说明	

2.25.150 LL_TIM_GetDMABurstBaseAddress

2.25.150.1 功能介绍

获取 DMA 基址。

2.25.150.2 接口定义

函数接口	uint32_t LL_TIM_GetDMABurstBaseAddress (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{ LL_TIM_DMABURST_BASEADDR_CR1, LL_TIM_DMABURST_BASEADDR_CR2, LL_TIM_DMABURST_BASEADDR_SMCR, LL_TIM_DMABURST_BASEADDR_DIER, LL_TIM_DMABURST_BASEADDR_SR, LL_TIM_DMABURST_BASEADDR_EGR, LL_TIM_DMABURST_BASEADDR_CCMR1, LL_TIM_DMABURST_BASEADDR_CCMR2, LL_TIM_DMABURST_BASEADDR_CCER, LL_TIM_DMABURST_BASEADDR_CNT, LL_TIM_DMABURST_BASEADDR_PSC, LL_TIM_DMABURST_BASEADDR_ARR, LL_TIM_DMABURST_BASEADDR_RCR, LL_TIM_DMABURST_BASEADDR_CCR1, LL_TIM_DMABURST_BASEADDR_CCR2, LL_TIM_DMABURST_BASEADDR_CCR3, LL_TIM_DMABURST_BASEADDR_CCR4, LL_TIM_DMABURST_BASEADDR_BDTR, LL_TIM_DMABURST_BASEADDR_OR1, LL_TIM_DMABURST_BASEADDR_CCMR3, LL_TIM_DMABURST_BASEADDR_CCR5, LL_TIM_DMABURST_BASEADDR_CCR6, LL_TIM_DMABURST_BASEADDR_AF1, LL_TIM_DMABURST_BASEADDR_AF2, LL_TIM_DMABURST_BASEADDR_TISEL }
资源使用	
说明	

2.25.151 LL_TIM_DMA_GetRegAddr

2.25.151.1 功能介绍

获取用于 DMA 传输的数据寄存器地址。

2.25.151.2 接口定义

函数接口	uint32_t LL_TIM_DMA_GetRegAddr (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.25.152 LL_TIM_DMA_GetRegARRAddr

2.25.152.1 功能介绍

获取用于 DMA 传输的 ARR 寄存器地址。

2.25.152.2 接口定义

函数接口	uint32_t LL_TIM_DMA_GetRegARRAddr (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.25.153 LL_TIM_DMA_GetRegCCR1Addr

2.25.153.1 功能介绍

获取用于 DMA 传输的 CCR1 寄存器地址。

2.25.153.2 接口定义

函数接口	uint32_t LL_TIM_DMA_GetRegCRR1Addr (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.25.154 LL_TIM_DMA_GetRegCCR2Addr

2.25.154.1 功能介绍

获取用于 DMA 传输的 CCR2 寄存器地址。

2.25.154.2 接口定义

函数接口	uint32_t LL_TIM_DMA_GetRegCRR2Addr (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	数据寄存器地址
资源使用	
说明	

2.25.155 LL_TIM_DMA_GetRegCCR3Addr

2.25.155.1 功能介绍

获取用于 DMA 传输的 CCR3 寄存器地址。

2.25.155.2 接口定义

函数接口	uint32_t LL_TIM_DMA_GetRegCRR3Addr (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.25.156 LL_TIM_DMA_GetRegCCR4Addr

2.25.156.1 功能介绍

获取用于 DMA 传输的 CCR4 寄存器地址。

2.25.156.2 接口定义

函数接口	uint32_t LL_TIM_DMA_GetRegCRR4Addr (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	数据寄存器地址
资源使用	
说明	

2.25.157 LL_TIM_SetRemap

2.25.157.1 功能介绍

重新映射 TIM 输入。

2.25.157.2 接口定义

函数接口	void LL_TIM_SetRemap (TIM_TypeDef * TIMx, uint32_t Remap)
输入	TIM_TypeDef * TIMx uint32_t Remap: { TIM1: LL_TIM_TIM1_TI1_RMP_GPIO, LL_TIM_TIM1_TI1_RMP_COMP1, LL_TIM_TIM1_TI2_RMP_GPIO, LL_TIM_TIM1_TI2_RMP_COMP2, LL_TIM_TIM1_TI3_RMP_GPIO, LL_TIM_TIM1_TI4_RMP_GPIO TIM2: LL_TIM_TIM2_TI1_RMP_GPIO, LL_TIM_TIM2_TI1_RMP_COMP1, LL_TIM_TIM2_TI2_RMP_GPIO, LL_TIM_TIM2_TI2_RMP_COMP2 TIM3: LL_TIM_TIM3_TI1_RMP_GPIO, LL_TIM_TIM3_TI1_RMP_COMP1,

	LL_TIM_TIM3_TI2_RMP_GPIO, LL_TIM_TIM3_TI2_RMP_COMP2 TIM4: LL_TIM_TIM4_TI1_RMP_GPIO, LL_TIM_TIM4_TI1_RMP_COMP1, LL_TIM_TIM4_TI2_RMP_GPIO, LL_TIM_TIM4_TI2_RMP_COMP2 }
输出	无
返回值	无
资源使用	
说明	

2.25.158 LL_TIM_SetChannelRemap

2.25.158.1 功能介绍

重新映射 TIM 输入(输入通道、内部/外部触发器)。

2.25.158.2 接口定义

函数接口	void LL_TIM_SetChannelRemap (TIM_TypeDef * TIMx, uint32_t Channel, uint32_t Remap)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 } uint32_t Remap: { TIM1: LL_TIM_TIM1_TI1_RMP_GPIO, LL_TIM_TIM1_TI1_RMP_COMP1, LL_TIM_TIM1_TI2_RMP_GPIO, LL_TIM_TIM1_TI2_RMP_COMP2, LL_TIM_TIM1_TI3_RMP_GPIO, LL_TIM_TIM1_TI4_RMP_GPIO TIM2: LL_TIM_TIM2_TI1_RMP_GPIO, LL_TIM_TIM2_TI1_RMP_COMP1, LL_TIM_TIM2_TI2_RMP_GPIO, LL_TIM_TIM2_TI2_RMP_COMP2 TIM3: LL_TIM_TIM3_TI1_RMP_GPIO, LL_TIM_TIM3_TI1_RMP_COMP1, LL_TIM_TIM3_TI2_RMP_GPIO, LL_TIM_TIM3_TI2_RMP_COMP2 TIM4: LL_TIM_TIM4_TI1_RMP_GPIO, LL_TIM_TIM4_TI1_RMP_COMP1, LL_TIM_TIM4_TI2_RMP_GPIO, LL_TIM_TIM4_TI2_RMP_COMP2 }
输出	无
返回值	无
资源使用	
说明	

2.25.159 LL_TIM_GetRemap

2.25.159.1 功能介绍

获取重新映射 TIM 输入。

2.25.159.2 接口定义

函数接口	uint32_t LL_TIM_GetRemap (TIM_TypeDef * TIMx)
------	---

输入	TIM_TypeDef * TIMx
输出	无
返回值	{ TIM1: LL_TIM_TIM1_TI1_RMP_GPIO, LL_TIM_TIM1_TI1_RMP_COMP1, LL_TIM_TIM1_TI2_RMP_GPIO, LL_TIM_TIM1_TI2_RMP_COMP2, LL_TIM_TIM1_TI3_RMP_GPIO, LL_TIM_TIM1_TI4_RMP_GPIO TIM2: LL_TIM_TIM2_TI1_RMP_GPIO, LL_TIM_TIM2_TI1_RMP_COMP1, LL_TIM_TIM2_TI2_RMP_GPIO, LL_TIM_TIM2_TI2_RMP_COMP2 TIM3: LL_TIM_TIM3_TI1_RMP_GPIO, LL_TIM_TIM3_TI1_RMP_COMP1, LL_TIM_TIM3_TI2_RMP_GPIO, LL_TIM_TIM3_TI2_RMP_COMP2 TIM4: LL_TIM_TIM4_TI1_RMP_GPIO, LL_TIM_TIM4_TI1_RMP_COMP1, LL_TIM_TIM4_TI2_RMP_GPIO, LL_TIM_TIM4_TI2_RMP_COMP2 }
资源使用	
说明	

2.25.160 LL_TIM_SetOCRefClearInputSource

2.25.160.1 功能介绍

设置 OCREF 清除输入源。

2.25.160.2 接口定义

函数接口	void LL_TIM_SetOCRefClearInputSource (TIM_TypeDef * TIMx, uint32_t OCRefClearInputSource)
输入	TIM_TypeDef * TIMx uint32_t OCRefClearInputSource: { LL_TIM_OCREF_CLR_INT_ETR, LL_TIM_OCREF_CLR_INT_COMP1, LL_TIM_OCREF_CLR_INT_COMP2 }
输出	无
返回值	无
资源使用	
说明	

2.25.161 LL_TIM_ClearFlag_UPDATE

2.25.161.1 功能介绍

清除更新中断标志。

2.25.161.2 接口定义

函数接口	void LL_TIM_ClearFlag_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	

说明	
----	--

2.25.162 LL_TIM_IsActiveFlag_UPDATE

2.25.162.1 功能介绍

检查更新中断标志。

2.25.162.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.163 LL_TIM_ClearFlag_CC1

2.25.163.1 功能介绍

清除捕获/比较 1 中断标志。

2.25.163.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.164 LL_TIM_IsActiveFlag_CC1

2.25.164.1 功能介绍

检查捕获/比较 1 中断标志。

2.25.164.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.165 LL_TIM_ClearFlag_CC2

2.25.165.1 功能介绍

清除捕获/比较 2 中断标志。

2.25.165.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC2(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx

输出	无
返回值	无
资源使用	
说明	

2.25.166 LL_TIM_IsActiveFlag_CC2

2.25.166.1 功能介绍

检查捕获/比较 2 中断标志。

2.25.166.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.167 LL_TIM_ClearFlag_CC3

2.25.167.1 功能介绍

清除捕获/比较 3 中断标志。

2.25.167.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.168 LL_TIM_IsActiveFlag_CC3

2.25.168.1 功能介绍

检查捕获/比较 3 中断标志。

2.25.168.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.169 LL_TIM_ClearFlag_CC4

2.25.169.1 功能介绍

清除捕获/比较 4 中断标志。

2.25.169.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.170 LL_TIM_IsActiveFlag_CC4

2.25.170.1 功能介绍

检查捕获/比较 4 中断标志。

2.25.170.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.171 LL_TIM_ClearFlag_CC5

2.25.171.1 功能介绍

清除捕获/比较 5 中断标志。

2.25.171.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC5 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.172 LL_TIM_IsActiveFlag_CC5

2.25.172.1 功能介绍

检查捕获/比较 5 中断标志。

2.25.172.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC5(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.173 LL_TIM_ClearFlag_CC6

2.25.173.1 功能介绍

清除捕获/比较 6 中断标志。

2.25.173.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC6 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.174 LL_TIM_IsActiveFlag_CC6

2.25.174.1 功能介绍

检查捕获/比较 6 中断标志。

2.25.174.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC6 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.175 LL_TIM_ClearFlag_COM

2.25.175.1 功能介绍

清除换向事件中断标志。

2.25.175.2 接口定义

函数接口	void LL_TIM_ClearFlag_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.176 LL_TIM_IsActiveFlag_COM

2.25.176.1 功能介绍

检查换向事件中断标志。

2.25.176.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	{0,1}
资源使用	
说明	

2.25.177 LL_TIM_ClearFlag_TRIG

2.25.177.1 功能介绍

清除触发中断标志。

2.25.177.2 接口定义

函数接口	void LL_TIM_ClearFlag_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.178 LL_TIM_IsActiveFlag_TRIG

2.25.178.1 功能介绍

检查触发中断标志。

2.25.178.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.179 LL_TIM_ClearFlag_BRK

2.25.179.1 功能介绍

清除断路中断标志。

2.25.179.2 接口定义

函数接口	void LL_TIM_ClearFlag_BRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.180 LL_TIM_IsActiveFlag_BRK

2.25.180.1 功能介绍

检查断路中断标志。

2.25.180.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_BRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.181 LL_TIM_ClearFlag_BRK2

2.25.181.1 功能介绍

清除断路 2 中断标志。

2.25.181.2 接口定义

函数接口	void LL_TIM_ClearFlag_BRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.182 LL_TIM_IsActiveFlag_BRK2

2.25.182.1 功能介绍

检查断路 2 中断标志。

2.25.182.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_BRK2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.183 LL_TIM_ClearFlag_CC1OVR

2.25.183.1 功能介绍

清除捕获/比较 1 重复捕获标志。

2.25.183.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC1OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.184 LL_TIM_IsActiveFlag_CC1OVR

2.25.184.1 功能介绍

检查捕获/比较 1 重复捕获标志。

2.25.184.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC1OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.185 LL_TIM_ClearFlag_CC2OVR

2.25.185.1 功能介绍

清除捕获/比较 2 重复捕获标志。

2.25.185.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC2OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.186 LL_TIM_IsActiveFlag_CC2OVR

2.25.186.1 功能介绍

检查捕获/比较 2 重复捕获标志。

2.25.186.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC2OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.187 LL_TIM_ClearFlag_CC3OVR

2.25.187.1 功能介绍

清除捕获/比较 3 重复捕获标志。

2.25.187.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC3OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	无
资源使用	
说明	

2.25.188 LL_TIM_IsActiveFlag_CC3OVR

2.25.188.1 功能介绍

检查捕获/比较 3 重复捕获标志。

2.25.188.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC3OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.189 LL_TIM_ClearFlag_CC4OVR

2.25.189.1 功能介绍

清除捕获/比较 4 重复捕获标志。

2.25.189.2 接口定义

函数接口	void LL_TIM_ClearFlag_CC4OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.190 LL_TIM_IsActiveFlag_CC4OVR

2.25.190.1 功能介绍

检查捕获/比较 4 重复捕获标志。

2.25.190.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_CC4OVR (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.191 LL_TIM_ClearFlag_SYSBRK

2.25.191.1 功能介绍

清除系统断路中断标志。

2.25.191.2 接口定义

函数接口	void LL_TIM_ClearFlag_SYSBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.192 LL_TIM_IsActiveFlag_SYSBRK

2.25.192.1 功能介绍

检查系统断路器中断标志。

2.25.192.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag_SYSBRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.193 LL_TIM_ClearFlag

2.25.193.1 功能介绍

清除中断标志。

2.25.193.2 接口定义

函数接口	void LL_TIM_ClearFlag (TIM_TypeDef * TIMx, uint32_t Flag)
输入	TIM_TypeDef * TIMx uint32_t Flag: { LL_TIM_SR_UIF, LL_TIM_SR_CC1IF, LL_TIM_SR_CC2IF, LL_TIM_SR_CC3IF, LL_TIM_SR_CC4IF, LL_TIM_SR_CC5IF, LL_TIM_SR_CC6IF, LL_TIM_SR_COMIF, LL_TIM_SR_TIF, LL_TIM_SR_BIF, LL_TIM_SR_B2IF, LL_TIM_SR_CC1OF, LL_TIM_SR_CC2OF, LL_TIM_SR_CC3OF, LL_TIM_SR_CC4OF, LL_TIM_SR_SBIF }
输出	无
返回值	无
资源使用	
说明	

2.25.194 LL_TIM_IsActiveFlag

2.25.194.1 功能介绍

检查中断标志。

2.25.194.2 接口定义

函数接口	uint32_t LL_TIM_IsActiveFlag (TIM_TypeDef * TIMx, uint32_t Flag)
输入	TIM_TypeDef * TIMx uint32_t Flag: { LL_TIM_SR_UIF, LL_TIM_SR_CC1IF, LL_TIM_SR_CC2IF, LL_TIM_SR_CC3IF, LL_TIM_SR_CC4IF, LL_TIM_SR_CC5IF, LL_TIM_SR_CC6IF, LL_TIM_SR_COMIF, LL_TIM_SR_TIF, LL_TIM_SR_BIF, LL_TIM_SR_B2IF, LL_TIM_SR_CC1OF, LL_TIM_SR_CC2OF, LL_TIM_SR_CC3OF, LL_TIM_SR_CC4OF, LL_TIM_SR_SBIF }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.195 LL_TIM_EnableIT_UPDATE

2.25.195.1 功能介绍

使能更新中断。

2.25.195.2 接口定义

函数接口	void LL_TIM_EnableIT_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.196 LL_TIM_DisableIT_UPDATE

2.25.196.1 功能介绍

禁止使能更新中断。

2.25.196.2 接口定义

函数接口	void LL_TIM_DisableIT_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.197 LL_TIM_IsEnabledIT_UPDATE

2.25.197.1 功能介绍

检查是否使能更新中断。

2.25.197.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.198 LL_TIM_EnableIT_CC1

2.25.198.1 功能介绍

使能捕获/比较 1 中断。

2.25.198.2 接口定义

函数接口	void LL_TIM_EnableIT_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.199 LL_TIM_DisableIT_CC1

2.25.199.1 功能介绍

禁止使能捕获/比较 1 中断。

2.25.199.2 接口定义

函数接口	void LL_TIM_DisableIT_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.200 LL_TIM_IsEnabledIT_CC1

2.25.200.1 功能介绍

检查是否使能捕获/比较 1 中断。

2.25.200.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.201 LL_TIM_EnableIT_CC2

2.25.201.1 功能介绍

使能捕获/比较 1 中断。

2.25.201.2 接口定义

函数接口	void LL_TIM_EnableIT_CC2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.202 LL_TIM_DisableIT_CC2

2.25.202.1 功能介绍

禁止使能捕获/比较 2 中断。

2.25.202.2 接口定义

函数接口	void LL_TIM_DisableIT_CC2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.203 LL_TIM_IsEnabledIT_CC2

2.25.203.1 功能介绍

检查是否使能捕获/比较 2 中断。

2.25.203.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_CC2(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.204 LL_TIM_EnableIT_CC3

2.25.204.1 功能介绍

使能捕获/比较 3 中断。

2.25.204.2 接口定义

函数接口	void LL_TIM_EnableIT_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	无
资源使用	
说明	

2.25.205 LL_TIM_DisableIT_CC3

2.25.205.1 功能介绍

禁止使能捕获/比较 3 中断。

2.25.205.2 接口定义

函数接口	void LL_TIM_DisableIT_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.206 LL_TIM_IsEnabledIT_CC3

2.25.206.1 功能介绍

检查是否使能捕获/比较 3 中断。

2.25.206.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.207 LL_TIM_EnableIT_CC4

2.25.207.1 功能介绍

使能捕获/比较 4 中断。

2.25.207.2 接口定义

函数接口	void LL_TIM_EnableIT_CC4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.208 LL_TIM_DisableIT_CC4

2.25.208.1 功能介绍

禁止使能捕获/比较 4 中断。

2.25.208.2 接口定义

函数接口	void LL_TIM_DisableIT_CC4(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.209 LL_TIM_IsEnabledIT_CC4

2.25.209.1 功能介绍

检查是否使能捕获/比较 4 中断。

2.25.209.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_CC4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.210 LL_TIM_EnableIT_COM

2.25.210.1 功能介绍

使能 COM 中断。

2.25.210.2 接口定义

函数接口	void LL_TIM_EnableIT_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.211 LL_TIM_DisableIT_COM

2.25.211.1 功能介绍

禁止使能 COM 中断。

2.25.211.2 接口定义

函数接口	void LL_TIM_DisableIT_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.212 LL_TIM_IsEnabledIT_COM

2.25.212.1 功能介绍

检查是否使能 COM 中断。

2.25.212.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.213 LL_TIM_EnableIT_TRIG

2.25.213.1 功能介绍

使能触发中断。

2.25.213.2 接口定义

函数接口	void LL_TIM_EnableIT_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.214 LL_TIM_DisableIT_TRIG

2.25.214.1 功能介绍

禁止使能触发中断。

2.25.214.2 接口定义

函数接口	void LL_TIM_DisableIT_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.215 LL_TIM_IsEnabledIT_TRIG

2.25.215.1 功能介绍

检查是否使能触发中断。

2.25.215.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	{0,1}
资源使用	
说明	

2.25.216 LL_TIM_EnableIT_BRK

2.25.216.1 功能介绍

使能断路中断。

2.25.216.2 接口定义

函数接口	void LL_TIM_EnableIT_BRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.217 LL_TIM_DisableIT_BRK

2.25.217.1 功能介绍

禁止使能断路中断。

2.25.217.2 接口定义

函数接口	void LL_TIM_DisableIT_BRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.218 LL_TIM_IsEnabledIT_BRK

2.25.218.1 功能介绍

检查是否使能断路中断。

2.25.218.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT_BRK (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.219 LL_TIM_EnableIT

2.25.219.1 功能介绍

使能中断。

2.25.219.2 接口定义

函数接口	void LL_TIM_EnableIT (TIM_TypeDef * TIMx, uint32_t Interrupt)
输入	TIM_TypeDef * TIMx uint32_t Interrupt: { LL_TIM_DIER_UIE, LL_TIM_DIER_CC1IE, LL_TIM_DIER_CC2IE, LL_TIM_DIER_CC3IE, LL_TIM_DIER_CC4IE, LL_TIM_DIER_COMIE, LL_TIM_DIER_TIE, LL_TIM_DIER_BIE }
输出	无
返回值	无
资源使用	
说明	

2.25.220 LL_TIM_DisableIT
2.25.220.1 功能介绍

禁止使能中断。

2.25.220.2 接口定义

函数接口	void LL_TIM_DisableIT (TIM_TypeDef * TIMx, uint32_t Interrupt)
输入	TIM_TypeDef * TIMx uint32_t Interrupt: { LL_TIM_DIER_UIE, LL_TIM_DIER_CC1IE, LL_TIM_DIER_CC2IE, LL_TIM_DIER_CC3IE, LL_TIM_DIER_CC4IE, LL_TIM_DIER_COMIE, LL_TIM_DIER_TIE, LL_TIM_DIER_BIE }
输出	无
返回值	无
资源使用	
说明	

2.25.221 LL_TIM_IsEnabledIT
2.25.221.1 功能介绍

检查是否使能中断。

2.25.221.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledIT (TIM_TypeDef * TIMx, uint32_t Interrupt)
输入	TIM_TypeDef * TIMx uint32_t Interrupt: { LL_TIM_DIER_UIE, LL_TIM_DIER_CC1IE, LL_TIM_DIER_CC2IE, LL_TIM_DIER_CC3IE, LL_TIM_DIER_CC4IE, LL_TIM_DIER_COMIE, LL_TIM_DIER_TIE, LL_TIM_DIER_BIE }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.222 LL_TIM_EnableDMAReq_UPDATE

2.25.222.1 功能介绍

使能更新 DMA 请求。

2.25.222.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.223 LL_TIM_DisableDMAReq_UPDATE

2.25.223.1 功能介绍

禁止使能更新 DMA 请求。

2.25.223.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.224 LL_TIM_IsEnabledDMAReq_UPDATE

2.25.224.1 功能介绍

检查是否使能更新 DMA 请求。

2.25.224.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.225 LL_TIM_DMAReq_CC1

2.25.225.1 功能介绍

使能捕获/比较 1DMA 请求。

2.25.225.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	无
资源使用	
说明	

2.25.226 LL_TIM_DisableDMAReq_CC1

2.25.226.1 功能介绍

禁止使能捕获/比较 1DMA 请求。

2.25.226.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.227 LL_TIM_IsEnabledDMAReq_CC1

2.25.227.1 功能介绍

检查是否使能捕获/比较 1DMA 请求。

2.25.227.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.228 LL_TIM_EnableDMAReq_CC2

2.25.228.1 功能介绍

使能捕获/比较 1DMA 请求。

2.25.228.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_CC2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.229 LL_TIM_DisableDMAReq_CC2

2.25.229.1 功能介绍

禁止使能捕获/比较 2DMA 请求。

2.25.229.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_CC2 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.230 LL_TIM_IsEnabledDMAReq_CC2
2.25.230.1 功能介绍

检查是否使能捕获/比较 2DMA 请求。

2.25.230.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_CC2(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.231 LL_TIM_EnableDMAReq_CC3
2.25.231.1 功能介绍

使能捕获/比较 3DMA 请求。

2.25.231.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.232 LL_TIM_DisableDMAReq_CC3
2.25.232.1 功能介绍

禁止使能捕获/比较 3DMA 请求。

2.25.232.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.233 LL_TIM_IsEnabledDMAReq_CC3

2.25.233.1 功能介绍

检查是否使能捕获/比较 3DMA 请求。

2.25.233.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_CC3 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.234 LL_TIM_EnableDMAReq_CC4

2.25.234.1 功能介绍

使能捕获/比较 4DMA 请求。

2.25.234.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_CC4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.235 LL_TIM_DisableDMAReq_CC4

2.25.235.1 功能介绍

禁止使能捕获/比较 4DMA 请求。

2.25.235.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_CC4(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.236 LL_TIM_IsEnabledDMAReq_CC4

2.25.236.1 功能介绍

检查是否使能捕获/比较 4DMA 请求。

2.25.236.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_CC4 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无

返回值	{0,1}
资源使用	
说明	

2.25.237 LL_TIM_EnableDMAReq_COM

2.25.237.1 功能介绍

使能 COM DMA 请求。

2.25.237.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.238 LL_TIM_DisableDMAReq_COM

2.25.238.1 功能介绍

禁止使能 COM DMA 请求。

2.25.238.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.239 LL_TIM_IsEnabledDMAReq_COM

2.25.239.1 功能介绍

检查是否使能 COM DMA 请求。

2.25.239.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_COM (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.240 LL_TIM_EnableDMAReq_TRIG

2.25.240.1 功能介绍

使能触发 DMA 请求。

2.25.240.2 接口定义

函数接口	void LL_TIM_EnableDMAReq_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.241 LL_TIM_DisableDMAReq_TRIG

2.25.241.1 功能介绍

禁止使能触发 DMA 请求。

2.25.241.2 接口定义

函数接口	void LL_TIM_DisableDMAReq_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.242 LL_TIM_IsEnabledDMAReq_TRIG

2.25.242.1 功能介绍

检查是否使能触发 DMA 请求。

2.25.242.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq_TRIG (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.243 LL_TIM_EnableDMAReq

2.25.243.1 功能介绍

使能 DMA 请求。

2.25.243.2 接口定义

函数接口	void LL_TIM_EnableDMAReq (TIM_TypeDef * TIMx, uint32_t Request)
输入	TIM_TypeDef * TIMx uint32_t Request: { LL_TIM_DMA_UPDATE, LL_TIM_DMA_CC1, LL_TIM_DMA_CC2, LL_TIM_DMA_CC3, LL_TIM_DMA_CC4, LL_TIM_DMA_COM, LL_TIM_DMA_TRIGGER }
输出	无

返回值	无
资源使用	
说明	

2.25.244 LL_TIM_DisableDMAReq

2.25.244.1 功能介绍

禁止使能 DMA 请求。

2.25.244.2 接口定义

函数接口	void LL_TIM_DisableDMAReq (TIM_TypeDef * TIMx, uint32_t Request)
输入	TIM_TypeDef * TIMx uint32_t Request: { LL_TIM_DMA_UPDATE, LL_TIM_DMA_CC1, LL_TIM_DMA_CC2, LL_TIM_DMA_CC3, LL_TIM_DMA_CC4, LL_TIM_DMA_COM, LL_TIM_DMA_TRIGGER }
输出	无
返回值	无
资源使用	
说明	

2.25.245 LL_TIM_IsEnabledDMAReq

2.25.245.1 功能介绍

检查是否使能 DMA 请求。

2.25.245.2 接口定义

函数接口	uint32_t LL_TIM_IsEnabledDMAReq (TIM_TypeDef * TIMx, uint32_t Request)
输入	TIM_TypeDef * TIMx uint32_t Request: { LL_TIM_DMA_UPDATE, LL_TIM_DMA_CC1, LL_TIM_DMA_CC2, LL_TIM_DMA_CC3, LL_TIM_DMA_CC4, LL_TIM_DMA_COM, LL_TIM_DMA_TRIGGER }
输出	无
返回值	{0,1}
资源使用	
说明	

2.25.246 LL_TIM_GenerateEvent_UPDATE

2.25.246.1 功能介绍

生成更新事件。

2.25.246.2 接口定义

函数接口	void LL_TIM_GenerateEvent_UPDATE (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx

输出	无
返回值	无
资源使用	
说明	

2.25.247 LL_TIM_GenerateEvent_CC1

2.25.247.1 功能介绍

生成捕获/比较 1 事件。

2.25.247.2 接口定义

函数接口	void LL_TIM_GenerateEvent_CC1 (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.248 LL_TIM_GenerateEvent_CC2

2.25.248.1 功能介绍

生成捕获/比较 2 事件。

2.25.248.2 接口定义

函数接口	void LL_TIM_GenerateEvent_CC2(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.249 LL_TIM_GenerateEvent_CC3

2.25.249.1 功能介绍

生成捕获/比较 3 事件。

2.25.249.2 接口定义

函数接口	void LL_TIM_GenerateEvent_CC3(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.250 LL_TIM_GenerateEvent_CC4

2.25.250.1 功能介绍

生成捕获/比较 4 事件。

2.25.250.2 接口定义

函数接口	void LL_TIM_GenerateEvent_CC4(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.251 LL_TIM_GenerateEvent_COM

2.25.251.1 功能介绍

生成 COM 事件。

2.25.251.2 接口定义

函数接口	void LL_TIM_GenerateEvent_COM(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.252 LL_TIM_GenerateEvent_TRIG

2.25.252.1 功能介绍

生成触发事件。

2.25.252.2 接口定义

函数接口	void LL_TIM_GenerateEvent_TRIG(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.253 LL_TIM_GenerateEvent_BRK

2.25.253.1 功能介绍

生成断路事件。

2.25.253.2 接口定义

函数接口	void LL_TIM_GenerateEvent_BRK(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.254 LL_TIM_GenerateEvent_BRK2

2.25.254.1 功能介绍

生成断路 2 事件。

2.25.254.2 接口定义

函数接口	void LL_TIM_GenerateEvent_BRK2(TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	无
资源使用	
说明	

2.25.255 LL_TIM_GenerateEvent

2.25.255.1 功能介绍

生成事件。

2.25.255.2 接口定义

函数接口	void LL_TIM_GenerateEvent (TIM_TypeDef * TIMx, uint32_t EventType)
输入	TIM_TypeDef * TIMx uint32_t EventType: { LL_TIM_EVENTSOURCE_UPDATE, LL_TIM_EVENTSOURCE_CC1, LL_TIM_EVENTSOURCE_CC2, LL_TIM_EVENTSOURCE_CC3, LL_TIM_EVENTSOURCE_CC4, LL_TIM_EVENTSOURCE_COM, LL_TIM_EVENTSOURCE_TRIGGER, LL_TIM_EVENTSOURCE_BREAK, LL_TIM_EVENTSOURCE_BREAK2 }
输出	无
返回值	无
资源使用	
说明	

2.25.256 LL_TIM_DeInit

2.25.256.1 功能介绍

清除 TIM 初始化参数。

2.25.256.2 接口定义

函数接口	ErrorStatus LL_TIM_DeInit (TIM_TypeDef * TIMx)
输入	TIM_TypeDef * TIMx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.25.257 LL_TIM_StructInit

2.25.257.1 功能介绍

TIM 结构体初始化。

2.25.257.2 接口定义

函数接口	void LL_TIM_StructInit (LL_TIM_InitTypeDef * TIM_InitStruct)
输入	LL_TIM_InitTypeDef * TIM_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.25.258 LL_TIM_Init

2.25.258.1 功能介绍

TIM 初始化。

2.25.258.2 接口定义

函数接口	ErrorStatus LL_TIM_Init (TIM_TypeDef * TIMx, LL_TIM_InitTypeDef * TIM_InitStruct)
输入	TIM_TypeDef * TIMx LL_TIM_InitTypeDef * TIM_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.25.259 LL_TIM_OC_StructInit

2.25.259.1 功能介绍

设置 TIMx 输出通道结构体初始化。

2.25.259.2 接口定义

函数接口	void LL_TIM_OC_StructInit (LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct)
输入	LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.25.260 LL_TIM_OC_Init

2.25.260.1 功能介绍

配置 TIMx 输出通道初始化。

2.25.260.2 接口定义

函数接口	ErrorStatus LL_TIM_OC_Init (TIM_TypeDef * TIMx, uint32_t Channel,
------	--

	LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4, LL_TIM_CHANNEL_CH5, LL_TIM_CHANNEL_CH6 } LL_TIM_OC_InitTypeDef * TIM_OC_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.25.261 LL_TIM_IC_StructInit

2.25.261.1 功能介绍

设置 TIMx 输入通道结构体初始化。

2.25.261.2 接口定义

函数接口	void LL_TIM_IC_StructInit (LL_TIM_IC_InitTypeDef * TIM_IC_InitStruct)
输入	LL_TIM_IC_InitTypeDef * TIM_IC_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.25.262 LL_TIM_IC_Init

2.25.262.1 功能介绍

配置 TIMx 输入通道初始化。

2.25.262.2 接口定义

函数接口	ErrorStatus LL_TIM_IC_Init (TIM_TypeDef * TIMx, uint32_t Channel, LL_TIM_IC_InitTypeDef * TIM_IC_InitStruct)
输入	TIM_TypeDef * TIMx uint32_t Channel: { LL_TIM_CHANNEL_CH1, LL_TIM_CHANNEL_CH2, LL_TIM_CHANNEL_CH3, LL_TIM_CHANNEL_CH4 } LL_TIM_IC_InitTypeDef * TIM_IC_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.25.263 LL_TIM_ENCODER_StructInit

2.25.263.1 功能介绍

编码器接口结构体初始化。

2.25.263.2 接口定义

函数接口	void LL_TIM_ENCODER_StructInit (LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct)
输入	LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct
输出	无
返回值	无
资源使用	
说明	

2.25.264 LL_TIM_ENCODER_Init

2.25.264.1 功能介绍

编码器接口配置初始化。

2.25.264.2 接口定义

函数接口	ErrorStatus LL_TIM_ENCODER_Init (TIM_TypeDef * TIMx, LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct)
输入	TIM_TypeDef * TIMx LL_TIM_ENCODER_InitTypeDef * TIM_EncoderInitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.25.265 LL_TIM_HALLSENSOR_StructInit

2.25.265.1 功能介绍

设置 TIMx 霍尔传感器接口配置数据结构体初始化。

2.25.265.2 接口定义

函数接口	void LL_TIM_HALLSENSOR_StructInit (LL_TIM_HALLSENSOR_InitTypeDef * TIM_HallSensorInitStruct)
输入	LL_TIM_HALLSENSOR_InitTypeDef * TIM_HallSensorInitStruct
输出	无
返回值	无
资源使用	
说明	

2.25.266 LL_TIM_HALLSENSOR_Init

2.25.266.1 功能介绍

配置定时器的霍尔传感器接口初始化。

2.25.266.2 接口定义

函数接口	ErrorStatus LL_TIM_HALLSENSOR_Init (TIM_TypeDef * TIMx, LL_TIM_HALLSENSOR_InitTypeDef * TIM_HallSensorInitStruct)
输入	TIM_TypeDef * TIMx LL_TIM_HALLSENSOR_InitTypeDef * TIM_HallSensorInitStruct

输出	无
返回值	ErrorStatus
资源使用	
说明	

2.25.267 LL_TIM_BDTR_StructInit

2.25.267.1 功能介绍

设置断路和死区配置结构体初始化。

2.25.267.2 接口定义

函数接口	void LL_TIM_BDTR_StructInit (LL_TIM_BDTR_InitTypeDef * TIM_BDTRInitStruct)
输入	LL_TIM_BDTR_InitTypeDef * TIM_BDTRInitStruct
输出	无
返回值	无
资源使用	
说明	

2.25.268 LL_TIM_BDTR_Init

2.25.268.1 功能介绍

设置断路和死区配置初始化。

2.25.268.2 接口定义

函数接口	ErrorStatus LL_TIM_BDTR_Init (TIM_TypeDef * TIMx, LL_TIM_BDTR_InitTypeDef * TIM_BDTRInitStruct)
输入	TIM_TypeDef * TIMx LL_TIM_BDTR_InitTypeDef * TIM_BDTRInitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.26 USART 模块

属性	类型	字段名	含义
USART_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	CR3	控制寄存器 3
读写	uint32_t	BRR	波特率分频寄存器
读写	uint32_t	GTPR	保护时间和预分频器寄存器

读写	uint32_t	RTOR	超时及块传输长度寄存器
只写	uint32_t	RQR	请求寄存器
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断标志清零寄存器
只读	uint32_t	RDR	接收数据寄存器
读写	uint32_t	TDR	发送数据寄存器
读写	uint32_t	PRESC	预分频器寄存器
LL_USART_InitTypeDef			
读写	uint32_t	PrescalerValue	波特率分频值
读写	uint32_t	BaudRate	波特率
读写	uint32_t	DataWidth	字符长度
读写	uint32_t	StopBits	停止位
读写	uint32_t	Parity	RS485 收发器使能信号极性
读写	uint32_t	TransferDirection	传输方向
读写	uint32_t	HardwareFlowControl	硬件流控制
读写	uint32_t	OverSampling	过采样模式
LL_USART_ClockInitTypeDef			
读写	uint32_t	ClockOutput	时钟输出
读写	uint32_t	ClockPolarity	时钟极性
读写	uint32_t	ClockPhase	时钟相位
读写	uint32_t	LastBitClockPulse	最后一位对应时钟脉冲

2.26.1 LL_USART_Enable

2.26.1.1 功能介绍

使能 USART。

2.26.1.2 接口定义

函数接口	void LL_USART_Enable (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.2 LL_USART_Disable

2.26.2.1 功能介绍

禁止使能 USART。

2.26.2.2 接口定义

函数接口	void LL_LPUART_Disable (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.3 LL_USART_IsEnabled

2.26.3.1 功能介绍

检查是否使能 USART。

2.26.3.2 接口定义

函数接口	uint32_t LL_USART_IsEnabled (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.4 LL_USART_EnableFIFO

2.26.4.1 功能介绍

使能 USART FIFO。

2.26.4.2 接口定义

函数接口	void LL_USART_EnableFIFO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.5 LL_USART_DisableFIFO

2.26.5.1 功能介绍

禁止使能 USART FIFO。

2.26.5.2 接口定义

函数接口	void LL_USART_DisableFIFO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无

返回值	无
资源使用	
说明	

2.26.6 LL_USART_IsEnabledFIFO

2.26.6.1 功能介绍

检查是否使能 USART FIFO。

2.26.6.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledFIFO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.7 LL_USART_SetTXFIFOThreshold

2.26.7.1 功能介绍

设置 USART 发送 FIFO 阈值。

2.26.7.2 接口定义

函数接口	void LL_USART_SetTXFIFOThreshold (USART_TypeDef * USARTx, uint32_t Threshold)
输入	USART_TypeDef * LUSARTx uint32_t Threshold: { LL_USART_FIFOTHRESHOLD_1_8, LL_USART_FIFOTHRESHOLD_1_4, LL_USART_FIFOTHRESHOLD_1_2, LL_USART_FIFOTHRESHOLD_3_4, LL_USART_FIFOTHRESHOLD_7_8, LL_USART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	无
资源使用	
说明	

2.26.8 LL_USART_GetTXFIFOThreshold

2.26.8.1 功能介绍

获取 USART 发送 FIFO 阈值。

2.26.8.2 接口定义

函数接口	uint32_t LL_USART_GetTXFIFOThreshold (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无

返回值	{ LL_USART_FIFOTHRESHOLD_1_8, LL_USART_FIFOTHRESHOLD_1_4, LL_USART_FIFOTHRESHOLD_1_2, LL_USART_FIFOTHRESHOLD_3_4, LL_USART_FIFOTHRESHOLD_7_8, LL_USART_FIFOTHRESHOLD_8_8 }
资源使用	
说明	

2.26.9 LL_USART_SetRXFIFOThreshold

2.26.9.1 功能介绍

设置 USART 接收 FIFO 阈值。

2.26.9.2 接口定义

函数接口	void LL_USART_SetRXFIFOThreshold (USART_TypeDef * USARTx, uint32_t Threshold)
输入	USART_TypeDef * USARTx uint32_t Threshold: { LL_USART_FIFOTHRESHOLD_1_8, LL_USART_FIFOTHRESHOLD_1_4, LL_USART_FIFOTHRESHOLD_1_2, LL_USART_FIFOTHRESHOLD_3_4, LL_USART_FIFOTHRESHOLD_7_8, LL_USART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	无
资源使用	
说明	

2.26.10 LL_USART_GetRXFIFOThreshold

2.26.10.1 功能介绍

获取 USART 接收 FIFO 阈值。

2.26.10.2 接口定义

函数接口	uint32_t LL_USART_GetRXFIFOThreshold (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_FIFOTHRESHOLD_1_8, LL_USART_FIFOTHRESHOLD_1_4, LL_USART_FIFOTHRESHOLD_1_2, LL_USART_FIFOTHRESHOLD_3_4, LL_USART_FIFOTHRESHOLD_7_8, LL_USART_FIFOTHRESHOLD_8_8 }
资源使用	

说明	
----	--

2.26.11 LL_USART_ConfigFIFOsThreshold

2.26.11.1 功能介绍

配置 USART 发送和接收 FIFO 阈值。

2.26.11.2 接口定义

函数接口	void LL_USART_ConfigFIFOsThreshold (USART_TypeDef * USARTx, uint32_t TXThreshold, uint32_t RXThreshold)
输入	USART_TypeDef * USARTx uint32_t TXThreshold: { LL_USART_FIFOTHRESHOLD_1_8, LL_USART_FIFOTHRESHOLD_1_4, LL_USART_FIFOTHRESHOLD_1_2, LL_USART_FIFOTHRESHOLD_3_4, LL_USART_FIFOTHRESHOLD_7_8, LL_USART_FIFOTHRESHOLD_8_8 } uint32_t RXThreshold: { LL_USART_FIFOTHRESHOLD_1_8, LL_USART_FIFOTHRESHOLD_1_4, LL_USART_FIFOTHRESHOLD_1_2, LL_USART_FIFOTHRESHOLD_3_4, LL_USART_FIFOTHRESHOLD_7_8, LL_USART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	
资源使用	
说明	

2.26.12 LL_USART_EnableInStopMode

2.26.12.1 功能介绍

使能 USART 将 MCU 从低功耗模式唤醒。

2.26.12.2 接口定义

函数接口	void LL_USART_EnableInStopMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.13 LL_USART_DisableInStopMode

2.26.13.1 功能介绍

禁止使能 USART 将 MCU 从低功耗模式唤醒。

2.26.13.2 接口定义

函数接口	void LL_LPUART_DisableInStopMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.14LL_USART_IsEnabledInStopMode

2.26.14.1 功能介绍

检查是否使能 USART 将 MCU 从低功耗模式唤醒。

2.26.14.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledInStopMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.15LL_USART_EnableDirectionRx

2.26.15.1 功能介绍

使能 USART 接收器使能。

2.26.15.2 接口定义

函数接口	void LL_USART_EnableDirectionRx (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.16LL_USART_DisableDirectionRx

2.26.16.1 功能介绍

禁止使能 USART 接收器使能。

2.26.16.2 接口定义

函数接口	void LL_USART_DisableDirectionRx (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.17 LL_USART_IsEnabledDirectionRx

2.26.17.1 功能介绍

检查是否使能 USART 接收器使能。

2.26.17.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledDirectionRx (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.18 LL_USART_EnableDirectionTx

2.26.18.1 功能介绍

使能 USART 发送器使能。

2.26.18.2 接口定义

函数接口	void LL_USART_EnableDirectionTx (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.19 LL_USART_DisableDirectionTx

2.26.19.1 功能介绍

禁止使能 USART 发送器使能。

2.26.19.2 接口定义

函数接口	void LL_USART_DisableDirectionTx (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.20 LL_USART_IsEnabledDirectionTx

2.26.20.1 功能介绍

检查是否使能 USART 发送器使能。

2.26.20.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledDirectionTx (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx

输出	无
返回值	{0,1}
资源使用	
说明	

2.26.21 LL_USART_SetTransferDirection

2.26.21.1 功能介绍

配置 USART 发送器、接收器的使能或禁止。

2.26.21.2 接口定义

函数接口	void LL_USART_SetTransferDirection (USART_TypeDef * USARTx, uint32_t TransferDirection)
输入	USART_TypeDef *USARTx uint32_t TransferDirection: { LL_USART_DIRECTION_NONE, LL_USART_DIRECTION_RX, LL_USART_DIRECTION_TX, LL_USART_DIRECTION_TX_RX }
输出	无
返回值	无
资源使用	
说明	

2.26.22 LL_USART_GetTransferDirection

2.26.22.1 功能介绍

获取 USART 发送器、接收器的使能或禁止。

2.26.22.2 接口定义

函数接口	uint32_t LL_USART_GetTransferDirection (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx uint32_t TransferDirection: { LL_USART_DIRECTION_NONE, LL_USART_DIRECTION_RX, LL_USART_DIRECTION_TX, LL_USART_DIRECTION_TX_RX }
输出	无
返回值	无
资源使用	
说明	

2.26.23 LL_USART_SetParity

2.26.23.1 功能介绍

配置 USART 奇偶校验。

2.26.23.2 接口定义

函数接口	void LL_USART_SetParity (USART_TypeDef * USARTx, uint32_t Parity)
输入	USART_TypeDef * USARTx uint32_t Parity:

	{ LL_USART_PARITY_NONE, LL_USART_PARITY_EVEN, LL_USART_PARITY_ODD }
输出	无
返回值	无
资源使用	
说明	

2.26.24 LL_USART_GetParity

2.26.24.1 功能介绍

获取 USART 奇偶校验。

2.26.24.2 接口定义

函数接口	uint32_t LL_USART_GetParity (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_PARITY_NONE, LL_USART_PARITY_EVEN, LL_USART_PARITY_ODD }
资源使用	
说明	

2.26.25 LL_USART_SetWakeUpMethod

2.26.25.1 功能介绍

配置 USART 唤醒方式。

2.26.25.2 接口定义

函数接口	void LL_USART_SetWakeUpMethod (USART_TypeDef * USARTx, uint32_t Method)
输入	USART_TypeDef * USARTx uint32_t Method: { LL_USART_WAKEUP_IDLELINE, LL_USART_WAKEUP_ADDRESSMARK }
输出	无
返回值	无
资源使用	
说明	

2.26.26 LL_USART_GetWakeUpMethod

2.26.26.1 功能介绍

获取 USART 唤醒方式。

2.26.26.2 接口定义

函数接口	uint32_t LL_USART_GetWakeUpMethod (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_WAKEUP_IDLELINE,

	LL_USART_WAKEUP_ADDRESSMARK }
资源使用	
说明	

2.26.27 LL_USART_SetDataWidth

2.26.27.1 功能介绍

配置 USART 字符长度。

2.26.27.2 接口定义

函数接口	void LL_USART_SetDataWidth (USART_TypeDef * USARTx, uint32_t DataWidth)
输入	USART_TypeDef * USARTx uint32_t DataWidth: { LL_USART_DATAWIDTH_7B, LL_USART_DATAWIDTH_8B, LL_USART_DATAWIDTH_9B }
输出	无
返回值	无
资源使用	
说明	

2.26.28 LL_USART_GetDataWidth

2.26.28.1 功能介绍

获取 USART 字符长度。

2.26.28.2 接口定义

函数接口	uint32_t LL_USART_GetDataWidth (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_DATAWIDTH_7B, LL_USART_DATAWIDTH_8B, LL_USART_DATAWIDTH_9B }
资源使用	
说明	

2.26.29 LL_USART_EnableMuteMode

2.26.29.1 功能介绍

使能静默模式。

2.26.29.2 接口定义

函数接口	void LL_USART_EnableMuteMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.30 LL_USART_DisableMuteMode

2.26.30.1 功能介绍

禁止使能静默模式。

2.26.30.2 接口定义

函数接口	void LL_USART_DisableMuteMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.31 LL_USART_IsEnabledMuteMode

2.26.31.1 功能介绍

检查是否使能静默模式。

2.26.31.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledMuteMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.32 LL_USART_SetOverSampling

2.26.32.1 功能介绍

配置 USART 过采样模式。

2.26.32.2 接口定义

函数接口	void LL_USART_SetOverSampling (USART_TypeDef * USARTx, uint32_t OverSampling)
输入	USART_TypeDef * USARTx uint32_t OverSampling: { LL_USART_OVERSAMPLING_16, LL_USART_OVERSAMPLING_8 }
输出	无
返回值	无
资源使用	
说明	

2.26.33 LL_USART_GetOverSampling

2.26.33.1 功能介绍

获取 USART 过采样模式。

2.26.33.2 接口定义

函数接口	uint32_t LL_USART_GetOverSampling (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_OVERSAMPLING_16, LL_USART_OVERSAMPLING_8 }
资源使用	
说明	

2.26.34LL_USART_SetLastClkPulseOutput

2.26.34.1 功能介绍

配置 USART 最后一位对应时钟脉冲。

2.26.34.2 接口定义

函数接口	void LL_USART_SetLastClkPulseOutput (USART_TypeDef * USARTx, uint32_t LastBitClockPulse)
输入	USART_TypeDef * USARTx uint32_t LastBitClockPulse: { LL_USART_LASTCLKPULSE_NO_OUTPUT, LL_USART_LASTCLKPULSE_OUTPUT }
输出	无
返回值	无
资源使用	
说明	

2.26.35LL_USART_GetLastClkPulseOutput

2.26.35.1 功能介绍

获取 USART 最后一位对应时钟脉冲。

2.26.35.2 接口定义

函数接口	uint32_t LL_USART_GetLastClkPulseOutput (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_LASTCLKPULSE_NO_OUTPUT, LL_USART_LASTCLKPULSE_OUTPUT }
资源使用	
说明	

2.26.36LL_USART_SetClockPhase

2.26.36.1 功能介绍

配置 USART 时钟相位。

2.26.36.2 接口定义

函数接口	void LL_USART_SetClockPhase (USART_TypeDef * USARTx, uint32_t
------	--

	ClockPhase)
输入	USART_TypeDef * USARTx uint32_t ClockPhase: {LL_USART_PHASE_1EDGE, LL_USART_PHASE_2EDGE }
输出	无
返回值	无
资源使用	
说明	

2.26.37LL_USART_GetClockPhase

2.26.37.1 功能介绍

获取 USART 时钟相位。

2.26.37.2 接口定义

函数接口	uint32_t LL_USART_GetClockPhase (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{LL_USART_PHASE_1EDGE, LL_USART_PHASE_2EDGE }
资源使用	
说明	

2.26.38LL_USART_SetClockPolarity

2.26.38.1 功能介绍

配置 USART 时钟极性。

2.26.38.2 接口定义

函数接口	void LL_USART_SetClockPolarity (USART_TypeDef * USARTx, uint32_t ClockPolarity)
输入	USART_TypeDef * USARTx uint32_t ClockPolarity: {LL_USART_POLARITY_LOW, LL_USART_POLARITY_HIGH }
输出	无
返回值	无
资源使用	
说明	

2.26.39LL_USART_GetClockPolarity

2.26.39.1 功能介绍

获取 USART 时钟极性。

2.26.39.2 接口定义

函数接口	uint32_t LL_USART_GetClockPolarity (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{LL_USART_POLARITY_LOW, LL_USART_POLARITY_HIGH }

资源使用	
说明	

2.26.40 LL_USART_ConfigClock

2.26.40.1 功能介绍

配置 USART 时钟。

2.26.40.2 接口定义

函数接口	void LL_USART_ConfigClock (USART_TypeDef * USARTx, uint32_t Phase, uint32_t Polarity, uint32_t LBCPOutput)
输入	USART_TypeDef * USARTx uint32_t Phase: {LL_USART_PHASE_1EDGE, LL_USART_PHASE_2EDGE } uint32_t Polarity: {LL_USART_POLARITY_LOW, LL_USART_POLARITY_HIGH } uint32_t LBCPOutput: { LL_USART_LASTCLKPULSE_NO_OUTPUT, LL_USART_LASTCLKPULSE_OUTPUT }
输出	无
返回值	无
资源使用	
说明	

2.26.41 LL_USART_SetPrescaler

2.26.41.1 功能介绍

设置时钟预分频。

2.26.41.2 接口定义

函数接口	void LL_USART_SetPrescaler (USART_TypeDef * USARTx, uint32_t PrescalerValue)
输入	USART_TypeDef * USARTx uint32_t PrescalerValue: { LL_USART_PRESCALER_DIV1, LL_USART_PRESCALER_DIV2, LL_USART_PRESCALER_DIV4, LL_USART_PRESCALER_DIV6, LL_USART_PRESCALER_DIV8, LL_USART_PRESCALER_DIV10, LL_USART_PRESCALER_DIV12, LL_USART_PRESCALER_DIV16, LL_USART_PRESCALER_DIV32, LL_LPUART_PRESCALER_DIV64, LL_USART_PRESCALER_DIV128, LL_USART_PRESCALER_DIV256 }
输出	无
返回值	无
资源使用	
说明	

2.26.42 LL_USART_GetPrescaler

2.26.42.1 功能介绍

获取时钟预分频。

2.26.42.2 接口定义

函数接口	uint32_t LL_USART_GetPrescaler (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_PRESCALER_DIV1, LL_USART_PRESCALER_DIV2, LL_USART_PRESCALER_DIV4, LL_USART_PRESCALER_DIV6, LL_USART_PRESCALER_DIV8, LL_USART_PRESCALER_DIV10, LL_USART_PRESCALER_DIV12, LL_USART_PRESCALER_DIV16, LL_USART_PRESCALER_DIV32, LL_LPUART_PRESCALER_DIV64, LL_USART_PRESCALER_DIV128, LL_USART_PRESCALER_DIV256 }
资源使用	
说明	

2.26.43 LL_USART_EnableSCLKOutput

2.26.43.1 功能介绍

使能时钟。

2.26.43.2 接口定义

函数接口	void LL_USART_EnableSCLKOutput (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.44 LL_USART_DisableSCLKOutput

2.26.44.1 功能介绍

禁止使能时钟。

2.26.44.2 接口定义

函数接口	void LL_USART_DisableSCLKOutput (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.45 LL_USART_IsEnabledSCLKOutput

2.26.45.1 功能介绍

检查是否使能时钟。

2.26.45.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledSCLKOutput (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.46 LL_USART_SetStopBitsLength

2.26.46.1 功能介绍

设置停止位长度。

2.26.46.2 接口定义

函数接口	void LL_USART_SetStopBitsLength (USART_TypeDef * USARTx, uint32_t StopBits)
输入	USART_TypeDef * USARTx uint32_t StopBits: { LL_USART_STOPBITS_0_5, LL_USART_STOPBITS_1, LL_USART_STOPBITS_1_5, LL_USART_STOPBITS_2 }
输出	无
返回值	无
资源使用	
说明	

2.26.47 LL_USART_GetStopBitsLength

2.26.47.1 功能介绍

获取停止位长度。

2.26.47.2 接口定义

函数接口	uint32_t LL_USART_GetStopBitsLength (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_STOPBITS_0_5, LL_USART_STOPBITS_1, LL_USART_STOPBITS_1_5, LL_USART_STOPBITS_2 }
资源使用	
说明	

2.26.48 LL_USART_ConfigCharacter

2.26.48.1 功能介绍

配置字符帧格式(数据宽度, 奇偶校验控制, 停止位)。

2.26.48.2 接口定义

函数接口	void LL_USART_ConfigCharacter (USART_TypeDef * USARTx, uint32_t DataWidth, uint32_t Parity, uint32_t StopBits)
------	--

输入	USART_TypeDef * LPUARTx uint32_t DataWidth: { LL_USART_DATAWIDTH_7B, LL_USART_DATAWIDTH_8B, LL_USART_DATAWIDTH_9B } uint32_t Parity: { LL_USART_PARITY_NONE, LL_USART_PARITY_EVEN, LL_USART_PARITY_ODD } uint32_t StopBits: { LL_USART_STOPBITS_0_5, LL_USART_STOPBITS_1, LL_USART_STOPBITS_1_5, LL_USART_STOPBITS_2 }
输出	无
返回值	无
资源使用	
说明	

2.26.49 LL_USART_SetTXRXSwap

2.26.49.1 功能介绍

设置 USARTTX/RX 引脚交换。

2.26.49.2 接口定义

函数接口	void LL_USART_SetTXRXSwap (USART_TypeDef * USARTx, uint32_t SwapConfig)
输入	USART_TypeDef * USARTx uint32_t SwapConfig: { LL_USART_TXRX_STANDARD, LL_USART_TXRX_SWAPPED }
输出	无
返回值	无
资源使用	
说明	

2.26.50 LL_USART_GetTXRXSwap

2.26.50.1 功能介绍

获取 USARTTX/RX 引脚交换。

2.26.50.2 接口定义

函数接口	uint32_t LL_USART_GetTXRXSwap (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_TXRX_STANDARD, LL_USART_TXRX_SWAPPED }
资源使用	
说明	

2.26.51 LL_USART_SetRXPinLevel

2.26.51.1 功能介绍

设置 USARTRX 引脚有效电平反向。

2.26.51.2 接口定义

函数接口	void LL_USART_SetRXPinLevel (USART_TypeDef * USARTx, uint32_t PinInvMethod)
输入	USART_TypeDef * USARTx uint32_t PinInvMethod: { LL_USART_RXPIN_LEVEL_STANDARD, LL_USART_RXPIN_LEVEL_INVERTED }
输出	无
返回值	无
资源使用	
说明	

2.26.52 LL_USART_GetRXPinLevel

2.26.52.1 功能介绍

获取 USARTRX 引脚有效电平反向。

2.26.52.2 接口定义

函数接口	uint32_t LL_USART_GetRXPinLevel (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_RXPIN_LEVEL_STANDARD, LL_USART_RXPIN_LEVEL_INVERTED }
资源使用	
说明	

2.26.53 LL_USART_SetTXPinLevel

2.26.53.1 功能介绍

设置 USARTTX 引脚有效电平反向。

2.26.53.2 接口定义

函数接口	void LL_USART_SetTXPinLevel (USART_TypeDef * USARTx, uint32_t PinInvMethod)
输入	USART_TypeDef * USARTx uint32_t PinInvMethod: { LL_USART_TXPIN_LEVEL_STANDARD, LL_USART_TXPIN_LEVEL_INVERTED }
输出	无
返回值	无
资源使用	
说明	

2.26.54 LL_USART_GetTXPinLevel

2.26.54.1 功能介绍

获取 USARTTX 引脚有效电平反向。

2.26.54.2 接口定义

函数接口	uint32_t LL_USART_GetTXPinLevel (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_TXPIN_LEVEL_STANDARD, LL_USART_TXPIN_LEVEL_INVERTED }
资源使用	
说明	

2.26.55 LL_USART_SetBinaryDataLogic

2.26.55.1 功能介绍

设置 USART 二进制数据极性反向。

2.26.55.2 接口定义

函数接口	void LL_USART_SetBinaryDataLogic (USART_TypeDef * USARTx, uint32_t DataLogic)
输入	USART_TypeDef * USARTx uint32_t DataLogic: { LL_USART_BINARY_LOGIC_POSITIVE, LL_USART_BINARY_LOGIC_NEGATIVE }
输出	无
返回值	无
资源使用	
说明	

2.26.56 LL_USART_GetBinaryDataLogic

2.26.56.1 功能介绍

获取 USART 二进制数据极性反向。

2.26.56.2 接口定义

函数接口	uint32_t LL_USART_GetBinaryDataLogic (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_BINARY_LOGIC_POSITIVE, LL_USART_BINARY_LOGIC_NEGATIVE }
资源使用	
说明	

2.26.57 LL_USART_SetTransferBitOrder

2.26.57.1 功能介绍

设置 USART MSB 是否优先。

2.26.57.2 接口定义

函数接口	void LL_LPUART_SetTransferBitOrder (USART_TypeDef * USARTx,
------	--

	uint32_t BitOrder)
输入	USART_TypeDef * USARTx uint32_t BitOrder: { LL_USART_BITORDER_LSBFIRST, LL_USART_BITORDER_MSBFIRST }
输出	无
返回值	无
资源使用	
说明	

2.26.58 LL_USART_GetTransferBitOrder

2.26.58.1 功能介绍

获取 USARTMSB 是否优先。

2.26.58.2 接口定义

函数接口	uint32_t LL_USART_GetTransferBitOrder (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_BITORDER_LSBFIRST, LL_USART_BITORDER_MSBFIRST }
资源使用	
说明	

2.26.59 LL_USART_EnableAutoBaudRate

2.26.59.1 功能介绍

使能自动波特率检测。

2.26.59.2 接口定义

函数接口	void LL_USART_EnableAutoBaudRate (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.60 LL_USART_DisableAutoBaudRate

2.26.60.1 功能介绍

禁止使能自动波特率检测。

2.26.60.2 接口定义

函数接口	void LL_USART_DisableAutoBaudRate (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无

资源使用	
说明	

2.26.61 LL_USART_IsEnabledAutoBaud

2.26.61.1 功能介绍

检查是否使能自动波特率检测。

2.26.61.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledAutoBaud (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.62 LL_USART_SetAutoBaudRateMode

2.26.62.1 功能介绍

设置 USART 自动波特率检测模式。

2.26.62.2 接口定义

函数接口	void LL_USART_SetAutoBaudRateMode (USART_TypeDef * USARTx, uint32_t AutoBaudRateMode)
输入	USART_TypeDef * USARTx uint32_t AutoBaudRateMode: { LL_USART_AUTOBAUD_DETECT_ON_STARTBIT, LL_USART_AUTOBAUD_DETECT_ON_FALLINGEDGE, LL_USART_AUTOBAUD_DETECT_ON_7F_FRAME, LL_USART_AUTOBAUD_DETECT_ON_55_FRAME }
输出	无
返回值	无
资源使用	
说明	

2.26.63 LL_USART_GetAutoBaudRateMode

2.26.63.1 功能介绍

获取 USART 自动波特率检测模式。

2.26.63.2 接口定义

函数接口	uint32_t LL_USART_GetAutoBaudRateMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_AUTOBAUD_DETECT_ON_STARTBIT, LL_USART_AUTOBAUD_DETECT_ON_FALLINGEDGE,

	LL_USART_AUTOBAUD_DETECT_ON_7F_FRAME, LL_USART_AUTOBAUD_DETECT_ON_55_FRAME }
资源使用	
说明	

2.26.64 LL_USART_EnableRxTimeout

2.26.64.1 功能介绍

使能接收超时。

2.26.64.2 接口定义

函数接口	void LL_USART_EnableRxTimeout (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.65 LL_USART_DisableRxTimeout

2.26.65.1 功能介绍

禁止使能接收超时。

2.26.65.2 接口定义

函数接口	void LL_USART_DisableRxTimeout (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.66 LL_USART_IsEnabledRxTimeout

2.26.66.1 功能介绍

检查是否使能接收超时。

2.26.66.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledRxTimeout (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.67 LL_USART_ConfigNodeAddress

2.26.67.1 功能介绍

设置 USART 本地节点地址。

2.26.67.2 接口定义

函数接口	void LL_USART_ConfigNodeAddress (USART_TypeDef * USARTx, uint32_t AddressLen, uint32_t NodeAddress)
输入	USART_TypeDef * USARTx uint32_t AddressLen: { LL_USART_ADDRESS_DETECT_4B, LL_USART_ADDRESS_DETECT_7B } uint32_t NodeAddress: [0, 255]
输出	无
返回值	无
资源使用	
说明	

2.26.68 LL_USART_GetNodeAddress

2.26.68.1 功能介绍

获取 USART 本地节点。

2.26.68.2 接口定义

函数接口	uint32_t LL_USART_GetNodeAddress (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0, 255]
资源使用	
说明	

2.26.69 LL_USART_GetNodeAddressLen

2.26.69.1 功能介绍

获取 USART 本地地址长度。

2.26.69.2 接口定义

函数接口	uint32_t LL_LPUSARTUART_GetNodeAddressLen (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_ADDRESS_DETECT_4B, LL_USART_ADDRESS_DETECT_7B }
资源使用	
说明	

2.26.70 LL_USART_EnableRTSHWFlowCtrl

2.26.70.1 功能介绍

使能 USARTRTS。

2.26.70.2 接口定义

函数接口	void LL_USART_EnableRTSHWFlowCtrl (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.71 LL_USART_DisableRTSHWFlowCtrl

2.26.71.1 功能介绍

禁止使能 USARTRTS。

2.26.71.2 接口定义

函数接口	void LL_USART_DisableRTSHWFlowCtrl (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.72 LL_USART_EnableCTSHWFlowCtrl

2.26.72.1 功能介绍

使能 USARTCTS。

2.26.72.2 接口定义

函数接口	void LL_USART_EnableCTSHWFlowCtrl (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.73 LL_USART_DisableCTSHWFlowCtrl

2.26.73.1 功能介绍

禁止使能 USARTCTS。

2.26.73.2 接口定义

函数接口	void LL_USART_DisableCTSHWFlowCtrl (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无

返回值	无
资源使用	
说明	

2.26.74 LL_USART_SetHWFlowCtrl

2.26.74.1 功能介绍

配置硬件流控制模式(CTS 和 RTS)。

2.26.74.2 接口定义

函数接口	void LL_USART_SetHWFlowCtrl (USART_TypeDef * USARTx, uint32_t HardwareFlowControl)
输入	USART_TypeDef * USARTx uint32_t HardwareFlowControl: { LL_USART_HWCONTROL_NONE, LL_USART_HWCONTROL_RTS, LL_USART_HWCONTROL_CTS, LL_USART_HWCONTROL_RTS_CTS, }
输出	无
返回值	无
资源使用	
说明	

2.26.75 LL_USART_GetHWFlowCtrl

2.26.75.1 功能介绍

获取硬件流控制模式(CTS 和 RTS)。

2.26.75.2 接口定义

函数接口	uint32_t LL_USART_GetHWFlowCtrl (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_HWCONTROL_NONE, LL_USART_HWCONTROL_RTS, LL_LPUART_HWCONTROL_CTS, LL_USART_HWCONTROL_RTS_CTS, }
资源使用	
说明	

2.26.76 LL_USART_EnableOneBitSamp

2.26.76.1 功能介绍

使能 USART 单次采样。

2.26.76.2 接口定义

函数接口	void LL_USART_EnableOneBitSamp (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	

说明	
----	--

2.26.77 LL_USART_DisableOneBitSamp

2.26.77.1 功能介绍

禁止使能 USART 单次采样。

2.26.77.2 接口定义

函数接口	void LL_USART_DisableOneBitSamp (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.78 LL_USART_IsEnabledOneBitSamp

2.26.78.1 功能介绍

检查是否使能 USART 单次采样。

2.26.78.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledOneBitSamp (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.79 LL_USART_EnableOverrunDetect

2.26.79.1 功能介绍

使能上溢检测。

2.26.79.2 接口定义

函数接口	void LL_USART_EnableOverrunDetect (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.80 LL_USART_DisableOverrunDetect

2.26.80.1 功能介绍

禁止使能上溢检测。

2.26.80.2 接口定义

函数接口	void LL_USART_DisableOverrunDetect (USART_TypeDef * USARTx)
------	---

输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.81 LL_USART_IsEnabledOverrunDetect

2.26.81.1 功能介绍

检查是否使能上溢检测。

2.26.81.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledOverrunDetect (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.82 LL_USART_SetWKUPType

2.26.82.1 功能介绍

设置唤醒中断的事件类型。

2.26.82.2 接口定义

函数接口	void LL_USART_SetWKUPType (USART_TypeDef * USARTx, uint32_t Type)
输入	USART_TypeDef * USARTx uint32_t Type: { LL_USART_WAKEUP_ON_ADDRESS, LL_USART_WAKEUP_ON_STARTBIT, LL_USART_WAKEUP_ON_RXNE }
输出	无
返回值	无
资源使用	
说明	

2.26.83 LL_USART_GetWKUPType

2.26.83.1 功能介绍

获取唤醒中断的事件类型。

2.26.83.2 接口定义

函数接口	uint32_t LL_USART_GetWKUPType (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_WAKEUP_ON_ADDRESS,

	LL_USART_WAKEUP_ON_STARTBIT, LL_USART_WAKEUP_ON_RXNE }
资源使用	
说明	

2.26.84 LL_USART_SetBaudRate

2.26.84.1 功能介绍

设置 USART 波特率。

2.26.84.2 接口定义

函数接口	void LL_USART_SetBaudRate (USART_TypeDef * USARTx, uint32_t PeriphClk, uint32_t PrescalerValue, uint32_t BaudRate)
输入	USART_TypeDef * USARTx uint32_t PeriphClk uint32_t PrescalerValue: { LL_USART_PRESCALER_DIV1, LL_USART_PRESCALER_DIV2, LL_USART_PRESCALER_DIV4, LL_USART_PRESCALER_DIV6, LL_USART_PRESCALER_DIV8, LL_USART_PRESCALER_DIV10 LL_USART_PRESCALER_DIV12, LL_USART_PRESCALER_DIV16, LL_USART_PRESCALER_DIV32, LL_USART_PRESCALER_DIV64, LL_USART_PRESCALER_DIV128, LL_USART_PRESCALER_DIV256 } uint32_t BaudRate
输出	无
返回值	无
资源使用	
说明	

2.26.85 LL_USART_GetBaudRate

2.26.85.1 功能介绍

获取 USART 波特率。

2.26.85.2 接口定义

函数接口	uint32_t LL_USART_GetBaudRate (USART_TypeDef * USARTx, uint32_t PeriphClk, uint32_t PrescalerValue)
输入	USART_TypeDef * USARTx uint32_t PeriphClk uint32_t PrescalerValue: { LL_USART_PRESCALER_DIV1, LL_USART_PRESCALER_DIV2, LL_USART_PRESCALER_DIV4, LL_USART_PRESCALER_DIV6, LL_USART_PRESCALER_DIV8, LL_USART_PRESCALER_DIV10 LL_USART_PRESCALER_DIV12, LL_USART_PRESCALER_DIV16, LL_USART_PRESCALER_DIV32, LL_USART_PRESCALER_DIV64, LL_USART_PRESCALER_DIV128, LL_USART_PRESCALER_DIV256 }
输出	无

返回值	波特率
资源使用	
说明	

2.26.86 LL_USART_SetRxTimeout

2.26.86.1 功能介绍

设置 USART 接收器超时时间。

2.26.86.2 接口定义

函数接口	void LL_USART_SetRxTimeout (USART_TypeDef * USARTx, uint32_t Timeout)
输入	USART_TypeDef * USARTx uint32_t Timeout: [0x00000000, 0x00FFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.26.87 LL_USART_GetRxTimeout

2.26.87.1 功能介绍

获取 USART 接收器超时时间。

2.26.87.2 接口定义

函数接口	uint32_t LL_USART_GetRxTimeout (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0x00000000, 0x00FFFFFF]
资源使用	
说明	

2.26.88 LL_USART_SetBlockLength

2.26.88.1 功能介绍

设置 USART 块长度。

2.26.88.2 接口定义

函数接口	void LL_USART_SetBlockLength (USART_TypeDef * USARTx, uint32_t BlockLength)
输入	USART_TypeDef * USARTx uint32_t BlockLength: [0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.26.89 LL_USART_GetBlockLength

2.26.89.1 功能介绍

获取 USART 长度。

2.26.89.2 接口定义

函数接口	uint32_t LL_USART_GetBlockLength (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.26.90 LL_USART_EnableIrda

2.26.90.1 功能介绍

使能 IrDA 模式。

2.26.90.2 接口定义

函数接口	void LL_USART_EnableIrda (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.91 LL_USART_DisableIrda

2.26.91.1 功能介绍

禁止使能 IrDA 模式。

2.26.91.2 接口定义

函数接口	void LL_USART_DisableIrda (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.92 LL_USART_IsEnabledIrda

2.26.92.1 功能介绍

检查是否使能 IrDA 模式。

2.26.92.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIrda (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无

返回值	{0,1}
资源使用	
说明	

2.26.93 LL_USART_SetIrdaPowerMode

2.26.93.1 功能介绍

设置 IrDA 模式。

2.26.93.2 接口定义

函数接口	void LL_USART_SetIrdaPowerMode (USART_TypeDef * USARTx, uint32_t PowerMode)
输入	USART_TypeDef * USARTx uint32_t PowerMode: { LL_USART_IRDA_POWER_NORMAL, LL_USART_IRDA_POWER_LOW }
输出	无
返回值	无
资源使用	
说明	

2.26.94 LL_USART_GetIrdaPowerMode

2.26.94.1 功能介绍

获取 IrDA 模式。

2.26.94.2 接口定义

函数接口	uint32_t LL_USART_GetIrdaPowerMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_IRDA_POWER_NORMAL, LL_USART_IRDA_POWER_LOW }
资源使用	
说明	

2.26.95 LL_USART_SetIrdaPrescaler

2.26.95.1 功能介绍

设置在 IrDA 低功耗和 IrDA 正常模式下预分频器值。

2.26.95.2 接口定义

函数接口	void LL_USART_SetIrdaPrescaler (USART_TypeDef * USARTx, uint32_t PrescalerValue)
输入	USART_TypeDef * USARTx uint32_t PrescalerValue: [0x00, 0xFF]
输出	无
返回值	无

资源使用	
说明	

2.26.96 LL_USART_GetIrdaPrescaler

2.26.96.1 功能介绍

获取在 IrDA 低功耗和 IrDA 正常模式下预分频器值。

2.26.96.2 接口定义

函数接口	uint32_t LL_USART_GetIrdaPrescaler (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.26.97 LL_USART_EnableHalfDuplex

2.26.97.1 功能介绍

使能单线半双工模式。

2.26.97.2 接口定义

函数接口	void LL_USART_EnableHalfDuplex (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.98 LL_USART_DisableHalfDuplex

2.26.98.1 功能介绍

禁止使能单线半双工模式。

2.26.98.2 接口定义

函数接口	void LL_USART_DisableHalfDuplex (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.99 LL_USART_IsEnabledHalfDuplex

2.26.99.1 功能介绍

检查是否使能单线半双工模式。

2.26.99.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledHalfDuplex (USART_TypeDef * USARTx)
------	--

	USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.100 LL_USART_EnableSPISlave

2.26.100.1 功能介绍

使能同步从模式。

2.26.100.2 接口定义

函数接口	void LL_USART_EnableSPISlave (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.101 LL_USART_DisableSPISlave

2.26.101.1 功能介绍

禁止使能同步从模式。

2.26.101.2 接口定义

函数接口	void LL_USART_DisableSPISlave (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.102 LL_USART_IsEnabledSPISlave

2.26.102.1 功能介绍

检查是否使能同步从模式。

2.26.102.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledSPISlave (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.103 LL_USART_EnableSPISlaveSelect

2.26.103.1 功能介绍

同步从设备一直处于被选中的状态，并忽略 NSS 引脚输入。

2.26.103.2 接口定义

函数接口	void LL_USART_EnableSPISlaveSelect (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.104 LL_USART_DisableSPISlaveSelect

2.26.104.1 功能介绍

同步从设备片选取决于 NSS 引脚输入。

2.26.104.2 接口定义

函数接口	void LL_USART_DisableSPISlaveSelect (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.105 LL_USART_IsEnabledSPISlaveSelect

2.26.105.1 功能介绍

检查当 NSS_SEL 位置 1 时，是否忽略 NSS 引脚输入。

2.26.105.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledSPISlaveSelect (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.106 LL_USART_SetDEDeassertionTime

2.26.106.1 功能介绍

RS485 模式下，设置禁止驱动器时 DE 信号的保持时间。

2.26.106.2 接口定义

函数接口	void LL_USART_SetDEDeassertionTime (USART_TypeDef * USARTx, uint32_t Time)
输入	USART_TypeDef * USARTx

	uint32_t Time: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.26.107 LL_USART_GetDEDeassertionTime

2.26.107.1 功能介绍

RS485 模式下，获取禁止驱动器时 DE 信号的保持时间。

2.26.107.2 接口定义

函数接口	uint32_t LL_USART_GetDEDeassertionTime (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0, 31]
资源使用	
说明	

2.26.108 LL_USART_SetDEAssertionTime

2.26.108.1 功能介绍

RS485 模式下，设置 DE 信号的建立时间。

2.26.108.2 接口定义

函数接口	void LL_USART_SetDEAssertionTime (USART_TypeDef * USARTx, uint32_t Time)
输入	USART_TypeDef * USARTx uint32_t Time: [0, 31]
输出	无
返回值	无
资源使用	
说明	

2.26.109 LL_USART_GetDEAssertionTime

2.26.109.1 功能介绍

RS485 模式下，获取 DE 信号的建立时间。

2.26.109.2 接口定义

函数接口	uint32_t LL_USART_GetDEAssertionTime (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0, 31]

资源使用	
说明	

2.26.110 LL_USART_EnableDEMode

2.26.110.1 功能介绍

使能 DE 功能。

2.26.110.2 接口定义

函数接口	void LL_USART_EnableDEMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.111 LL_USART_DisableDEMode

2.26.111.1 功能介绍

禁止使能 DE 功能。

2.26.111.2 接口定义

函数接口	void LL_USART_DisableDEMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.112 LL_USART_IsEnabledDEMode

2.26.112.1 功能介绍

检查是否使能 DE 功能。

2.26.112.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledDEMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.113 LL_USART_SetDESignalPolarity

2.26.113.1 功能介绍

设置 RS485 收发器使能信号极性。

2.26.113.2 接口定义

函数接口	void LL_USART_SetDESignalPolarity (USART_TypeDef * USARTx,
------	---

	uint32_t Polarity)
输入	USART_TypeDef * USARTx uint32_t Polarity: { LL_USART_DE_POLARITY_HIGH, LL_USART_DE_POLARITY_LOW }
输出	无
返回值	无
资源使用	
说明	

2.26.114 LL_USART_GetDESignalPolarity

2.26.114.1 功能介绍

获取 RS485 收发器使能信号极性。

2.26.114.2 接口定义

函数接口	uint32_t LL_USART_GetDESignalPolarity (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{ LL_USART_DE_POLARITY_HIGH, LL_USART_DE_POLARITY_LOW }
资源使用	
说明	

2.26.115 LL_USART_ConfigAsyncMode

2.26.115.1 功能介绍

执行 USART 的基本配置，使其能够在异步模式下使用。

2.26.115.2 接口定义

函数接口	void LL_USART_ConfigAsyncMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.116 LL_USART_ConfigSyncMode

2.26.116.1 功能介绍

执行 USART 的基本配置，使其能够在同步模式下使用。

2.26.116.2 接口定义

函数接口	void LL_USART_ConfigSyncMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无

资源使用	
说明	

2.26.117 LL_USART_ConfigHalfDuplexMode

2.26.117.1 功能介绍

执行 USART 的基本配置，使其能够在半双工模式下使用。

2.26.117.2 接口定义

函数接口	void LL_USART_ConfigHalfDuplexMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.118 LL_USART_ConfigIrdaMode

2.26.118.1 功能介绍

执行 USART 的基本配置，使其能够在 IRDA 模式下使用。

2.26.118.2 接口定义

函数接口	void LL_USART_ConfigIrdaMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.119 LL_USART_ConfigMultiProcessMode

2.26.119.1 功能介绍

执行 USART 的基本配置，使其能够在多处理器模式下使用。

2.26.119.2 接口定义

函数接口	void LL_USART_ConfigIrdaMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.120 LL_USART_IsActiveFlag_PE

2.26.120.1 功能介绍

检查奇偶校验错误标志。

2.26.120.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_PE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.121 LL_USART_IsActiveFlag_FE

2.26.121.1 功能介绍

检查帧错误标志。

2.26.121.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_FE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.122 LL_USART_IsActiveFlag_NE

2.26.122.1 功能介绍

检查噪声检测标志。

2.26.122.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_NE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.123 LL_USART_IsActiveFlag_ORE

2.26.123.1 功能介绍

检查上溢错误标志。

2.26.123.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_ORE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.124 LL_USART_IsActiveFlag_IDLE

2.26.124.1 功能介绍

检查检测到空闲帧标志。

2.26.124.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_IDLE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.125 LL_USART_IsActiveFlag_RXNE_RXFNE

2.26.125.1 功能介绍

检查 RXFIFO 非空标志。

2.26.125.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_RXNE_RXFNE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.126 LL_USART_IsActiveFlag_TC

2.26.126.1 功能介绍

检查发送完成标志。

2.26.126.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_TC (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.127 LL_USART_IsActiveFlag_TXE_TXFNF

2.26.127.1 功能介绍

检查 TXFIFO 未满足标志。

2.26.127.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_TXE_TXFNF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx

输出	无
返回值	{0,1}
资源使用	
说明	

2.26.128 LL_USART_IsActiveFlag_nCTS

2.26.128.1 功能介绍

检查 CTS 中断标志。

2.26.128.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_nCTS (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.129 LL_USART_IsActiveFlag_CTS

2.26.129.1 功能介绍

检查 CTS 标志。

2.26.129.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_CTS (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.130 LL_USART_IsActiveFlag_RTO

2.26.130.1 功能介绍

检查接收器超时标志。

2.26.130.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_RTO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.131 LL_USART_IsActiveFlag_UDR

2.26.131.1 功能介绍

检查同步从设备下溢出错误标志。

2.26.131.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_UDR (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.132 LL_USART_IsActiveFlag_ABRE

2.26.132.1 功能介绍

检查自动波特率检测错误标志。

2.26.132.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_ABRE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.133 LL_USART_IsActiveFlag_ABR

2.26.133.1 功能介绍

检查自动波特率检测标志。

2.26.133.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_ABR (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.134 LL_USART_IsActiveFlag_BUSY

2.26.134.1 功能介绍

检查忙标志。

2.26.134.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_BUSY (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	

说明	
----	--

2.26.135 LL_USART_IsActiveFlag_CM

2.26.135.1 功能介绍

检查字符匹配标志。

2.26.135.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_CM (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.136 LL_USART_IsActiveFlag_SBK

2.26.136.1 功能介绍

检查中断帧发送标志。

2.26.136.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_SBK (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.137 LL_USART_IsActiveFlag_RWU

2.26.137.1 功能介绍

检查静默模式状态指示位。

2.26.137.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_RWU (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.138 LL_USART_IsActiveFlag_WKUP

2.26.138.1 功能介绍

检查从低功耗模式唤醒标志。

2.26.138.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_WKUP (USART_TypeDef * USARTx)
------	--

输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.139 LL_USART_IsActiveFlag_TEACK

2.26.139.1 功能介绍

检查发送使能确认标志。

2.26.139.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_TEACK (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.140 LL_USART_IsActiveFlag_REACK

2.26.140.1 功能介绍

检查接收使能确认标志。

2.26.140.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_REACK (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.141 LL_USART_IsActiveFlag_TXFE

2.26.141.1 功能介绍

检查 TXFIFO 空标志。

2.26.141.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_TXFE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.142 LL_USART_IsActiveFlag_RXFE

2.26.142.1 功能介绍

检查 RX FIFO 已满标志。

2.26.142.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_RXFE(USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.143 LL_USART_IsActiveFlag_TXFT

2.26.143.1 功能介绍

检查 TX FIFO 阈值标志。

2.26.143.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_TXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.144 LL_USART_IsActiveFlag_RXFT

2.26.144.1 功能介绍

检查 RX FIFO 阈值标志。

2.26.144.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag_RXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.145 LL_USART_IsActiveFlag

2.26.145.1 功能介绍

检查标志。

2.26.145.2 接口定义

函数接口	uint32_t LL_USART_IsActiveFlag (USART_TypeDef * USARTx, uint32_t Flag)
------	--

输入	USART_TypeDef * USARTx uint32_t Flag: { LL_USART_ISR_FE, LL_USART_ISR_NE, LL_USART_ISR_ORE, LL_USART_ISR_IDLE, LL_USART_ISR_RXNE_RXFNE, LL_USART_ISR_TC, LL_USART_ISR_TXE_TXFNF, LL_USART_ISR_CTSIF, LL_USART_ISR_CTS, LL_USART_ISR_RTOF, LL_USART_ISR_UDR, LL_USART_ISR_ABRE, LL_USART_ISR_ABRF, LL_USART_ISR_BUSY, LL_USART_ISR_CMF, LL_USART_ISR_SBKF, LL_USART_ISR_RWU, LL_USART_ISR_WUF, LL_USART_ISR_TEACK, LL_USART_ISR_REACK, LL_USART_ISR_TXFE, LL_USART_ISR_RXFF, LL_USART_ISR_RXFT, LL_USART_ISR_TXFT }
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.146 LL_USART_ClearFlag_PE

2.26.146.1 功能介绍

清除奇偶校验错误标志。

2.26.146.2 接口定义

函数接口	void LL_USART_ClearFlag_PE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.147 LL_USART_ClearFlag_FE

2.26.147.1 功能介绍

清除帧错误标志。

2.26.147.2 接口定义

函数接口	void LL_USART_ClearFlag_FE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.148 LL_USART_ClearFlag_NE

2.26.148.1 功能介绍

清除噪声检测标志。

2.26.148.2 接口定义

函数接口	void LL_USART_ClearFlag_NE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.149 LL_USART_ClearFlag_ORE

2.26.149.1 功能介绍

清除上溢错误标志。

2.26.149.2 接口定义

函数接口	void LL_USART_ClearFlag_PE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.150 LL_USART_ClearFlag_IDLE

2.26.150.1 功能介绍

清除空闲线路标志。

2.26.150.2 接口定义

函数接口	void LL_USART_ClearFlag_IDLE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.151 LL_USART_ClearFlag_TC

2.26.151.1 功能介绍

清除发送完成标志。

2.26.151.2 接口定义

函数接口	void LL_USART_ClearFlag_TC (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无

返回值	无
资源使用	
说明	

2.26.152 LL_USART_ClearFlag_nCTS

2.26.152.1 功能介绍

清除 CTS 标志。

2.26.152.2 接口定义

函数接口	void LL_USART_ClearFlag_nCTS (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.153 LL_USART_ClearFlag_CM

2.26.153.1 功能介绍

清除字符匹配标志。

2.26.153.2 接口定义

函数接口	void LL_USART_ClearFlag_CM (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.154 LL_USART_ClearFlag_WKUP

2.26.154.1 功能介绍

清除从低功耗模式唤醒标志。

2.26.154.2 接口定义

函数接口	void LL_USART_ClearFlag_WKUP (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.155 LL_USART_ClearFlag_RTO

2.26.155.1 功能介绍

清除接收超时标志。

2.26.155.2 接口定义

函数接口	void LL_USART_ClearFlag_RTO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.156 LL_USART_ClearFlag_UDR

2.26.156.1 功能介绍

清除下溢标志。

2.26.156.2 接口定义

函数接口	void LL_USART_ClearFlag_UDR (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.157 LL_USART_ClearFlag

2.26.157.1 功能介绍

清除 USART 标志。

2.26.157.2 接口定义

函数接口	void LL_USART_ClearFlag (USART_TypeDef * USARTx, uint32_t Flag)
输入	USART_TypeDef * USARTx uint32_t Flag: { LL_USART_ICR_PECF, LL_USART_ICR_FECF, LL_USART_ICR_NECF, LL_USART_ICR_ORECF, LL_USART_ICR_IDLECF, LL_USART_ICR_TCCF, LL_USART_ICR_CTSCF, LL_USART_ICR_RTOCF, LL_USART_ICR_UDRCF, LL_USART_ICR_CMCF, LL_USART_ICR_WUCF }
输出	无
返回值	无
资源使用	
说明	

2.26.158 LL_USART_EnableIT_IDLE

2.26.158.1 功能介绍

使能空闲帧检测中断。

2.26.158.2 接口定义

函数接口	void LL_USART_EnableIT_IDLE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.159 LL_USART_EnableIT_RXNE_RXFNE

2.26.159.1 功能介绍

使能接收数据非空中断。

2.26.159.2 接口定义

函数接口	void LL_USART_EnableIT_RXNE_RXFNE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.160 LL_USART_EnableIT_TC

2.26.160.1 功能介绍

使能传输完成中断。

2.26.160.2 接口定义

函数接口	void LL_USART_EnableIT_TC (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.161 LL_USART_EnableIT_TXE_TXFNF

2.26.161.1 功能介绍

使能数据寄存器空中断。

2.26.161.2 接口定义

函数接口	void LL_USART_EnableIT_TXE_TXFNF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	

说明	
----	--

2.26.162 LL_USART_EnableIT_PE

2.26.162.1 功能介绍

使能奇偶校验错误中断。

2.26.162.2 接口定义

函数接口	void LL_USART_EnableIT_PE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.163 LL_USART_EnableIT_CM

2.26.163.1 功能介绍

使能字符匹配中断。

2.26.163.2 接口定义

函数接口	void LL_USART_EnableIT_CM (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.164 LL_USART_EnableIT_RTO

2.26.164.1 功能介绍

使能接收超时中断。

2.26.164.2 接口定义

函数接口	void LL_USART_EnableIT_RTO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.165 LL_USART_EnableIT_TXFE

2.26.165.1 功能介绍

使能发送 FIFO 阈值中断。

2.26.165.2 接口定义

函数接口	void LL_USART_EnableIT_TXFE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx

输出	无
返回值	无
资源使用	
说明	

2.26.166 LL_USART_EnableIT_RXFF

2.26.166.1 功能介绍

使能接收 FIFO 阈值中断。

2.26.166.2 接口定义

函数接口	void LL_USART_EnableIT_RXFF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.167 LL_USART_EnableIT_ERROR

2.26.167.1 功能介绍

使能错误中断。

2.26.167.2 接口定义

函数接口	void LL_USART_EnableIT_ERROR (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.168 LL_USART_EnableIT_CTS

2.26.168.1 功能介绍

使能 CTS 中断。

2.26.168.2 接口定义

函数接口	void LL_USART_EnableIT_CTS (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.169 LL_USART_EnableIT_WKUP

2.26.169.1 功能介绍

使能从低功耗模式唤醒信号中断。

2.26.169.2 接口定义

函数接口	void LL_USART_EnableIT_WKUP (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.170 LL_USART_EnableIT_TXFT

2.26.170.1 功能介绍

使能发送 FIFO 阈值中断。

2.26.170.2 接口定义

函数接口	void LL_USART_EnableIT_TXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.171 LL_USART_EnableIT_RXFT

2.26.171.1 功能介绍

使能接收 FIFO 阈值中断。

2.26.171.2 接口定义

函数接口	void LL_USART_EnableIT_RXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.172 LL_USART_DisableIT_IDLE

2.26.172.1 功能介绍

禁止使能空闲帧检测中断。

2.26.172.2 接口定义

函数接口	void LL_USART_DisableIT_IDLE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.173 LL_USART_DisableIT_RXNE_RXFNE

2.26.173.1 功能介绍

禁止使能接收数据非空中断。

2.26.173.2 接口定义

函数接口	void LL_USART_DisableIT_RXNE_RXFNE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.174 LL_USART_DisableIT_TC

2.26.174.1 功能介绍

禁止使能传输完成中断。

2.26.174.2 接口定义

函数接口	void LL_USART_DisableIT_TC (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.175 LL_USART_DisableIT_TXE_TXFNF

2.26.175.1 功能介绍

禁止使能数据寄存器空中断。

2.26.175.2 接口定义

函数接口	void LL_USART_DisableIT_TXE_TXFNF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.176 LL_USART_DisableIT_PE

2.26.176.1 功能介绍

禁止使能奇偶校验错误中断。

2.26.176.2 接口定义

函数接口	void LL_USART_DisableIT_PE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx

输出	无
返回值	无
资源使用	
说明	

2.26.177 LL_USART_DisableIT_CM

2.26.177.1 功能介绍

禁止使能字符匹配中断。

2.26.177.2 接口定义

函数接口	void LL_USART_DisableIT_CM (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.178 LL_USART_DisableIT_RTO

2.26.178.1 功能介绍

禁止使能接收超时中断。

2.26.178.2 接口定义

函数接口	void LL_USART_DisableIT_RTO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.179 LL_USART_DisableIT_TXFE

2.26.179.1 功能介绍

禁止使能发送 FIFO 阈值中断。

2.26.179.2 接口定义

函数接口	void LL_USART_DisableIT_TXFE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.180 LL_USART_DisableIT_RXFF

2.26.180.1 功能介绍

禁止使能接收 FIFO 阈值中断。

2.26.180.2 接口定义

函数接口	void LL_USART_DisableIT_RXFF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.181 LL_USART_DisableIT_ERROR

2.26.181.1 功能介绍

禁止使能错误中断。

2.26.181.2 接口定义

函数接口	void LL_USART_DisableIT_ERROR (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.182 LL_USART_DisableIT_CTS

2.26.182.1 功能介绍

禁止使能 CTS 中断。

2.26.182.2 接口定义

函数接口	void LL_USART_DisableIT_CTS (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.183 LL_USART_DisableIT_WKUP

2.26.183.1 功能介绍

禁止使能从低功耗模式唤醒信号中断。

2.26.183.2 接口定义

函数接口	void LL_USART_DisableIT_WKUP (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.184 LL_USART_DisableIT_TXFT

2.26.184.1 功能介绍

禁止使能发送 FIFO 阈值中断。

2.26.184.2 接口定义

函数接口	void LL_USART_DisableIT_TXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.185 LL_USART_DisableIT_RXFT

2.26.185.1 功能介绍

禁止使能接收 FIFO 阈值中断。

2.26.185.2 接口定义

函数接口	void LL_USART_DisableIT_RXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.186 LL_USART_IsEnabledIT_IDLE

2.26.186.1 功能介绍

检查是否使能空闲帧检测中断。

2.26.186.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_IDLE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.187 LL_USART_IsEnabledIT_RXNE_RXFNE

2.26.187.1 功能介绍

检查是否使能接收数据非空中断。

2.26.187.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_RXNE_RXFNE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx

输出	无
返回值	{0,1}
资源使用	
说明	

2.26.188 LL_USART_IsEnabledIT_TC

2.26.188.1 功能介绍

检查是否使能传输完成中断。

2.26.188.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_TC (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.189 LL_USART_IsEnabledIT_TXE_TXFNF

2.26.189.1 功能介绍

检查是否使能数据寄存器空中断。

2.26.189.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_TXE_TXFNF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.190 LL_USART_IsEnabledIT_PE

2.26.190.1 功能介绍

检查是否使能奇偶校验错误中断。

2.26.190.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_PE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.191 LL_USART_IsEnabledIT_CM

2.26.191.1 功能介绍

检查是否使能字符匹配中断。

2.26.191.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_CM (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.192 LL_USART_IsEnabledIT_RTO

2.26.192.1 功能介绍

检查是否使能接收超时中断。

2.26.192.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_RTO (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.193 LL_USART_IsEnabledIT_TXFE

2.26.193.1 功能介绍

检查是否使能发送 FIFO 阈值中断。

2.26.193.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_TXFE (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.194 LL_USART_IsEnabledIT_RXFF

2.26.194.1 功能介绍

检查是否使能接收 FIFO 阈值中断。

2.26.194.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_RXFF (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.195 LL_USART_IsEnabledIT_ERROR

2.26.195.1 功能介绍

检查是否使能错误中断。

2.26.195.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_ERROR (USART_TypeDef * USARTx)
输入	USART_TypeDef *USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.196 LL_USART_IsEnabledIT_CTS

2.26.196.1 功能介绍

检查是否使能 CTS 中断。

2.26.196.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_CTS (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.197 LL_USART_IsEnabledIT_WKUP

2.26.197.1 功能介绍

检查是否使能从低功耗模式唤醒信号中断。

2.26.197.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_WKUP (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.198 LL_USART_IsEnabledIT_TXFT

2.26.198.1 功能介绍

检查是否使能发送 FIFO 阈值中断。

2.26.198.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_TXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx

输出	无
返回值	{0,1}
资源使用	
说明	

2.26.199 LL_USART_IsEnabledIT_RXFT

2.26.199.1 功能介绍

检查是否使能接收 FIFO 阈值中断。

2.26.199.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledIT_RXFT (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.200 LL_USART_EnableDMAReq_RX

2.26.200.1 功能介绍

使能 DMA 模式接收。

2.26.200.2 接口定义

函数接口	void LL_USART_EnableDMAReq_RX (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.201 LL_USART_DisableDMAReq_RX

2.26.201.1 功能介绍

禁止使能 DMA 模式接收。

2.26.201.2 接口定义

函数接口	void LL_USART_DisableDMAReq_RX (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.202 LL_USART_IsEnabledDMAReq_RX

2.26.202.1 功能介绍

使能 DMA 模式接收。

2.26.202.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledDMAReq_RX (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	
说明	

2.26.203 LL_USART_EnableDMAReq_TX

2.26.203.1 功能介绍

使能 DMA 模式发送。

2.26.203.2 接口定义

函数接口	void LL_USART_EnableDMAReq_TX (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.204 LL_USART_DisableDMAReq_TX

2.26.204.1 功能介绍

禁止使能 DMA 模式发送。

2.26.204.2 接口定义

函数接口	void LL_USART_DisableDMAReq_TX (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.205 LL_USART_IsEnabledDMAReq_TX

2.26.205.1 功能介绍

使能 DMA 模式发送。

2.26.205.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledDMAReq_TX (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	{0,1}
资源使用	

说明	
----	--

2.26.206 LL_USART_EnableDMADeactOnRxErr

2.26.206.1 功能介绍

接收出错时禁止 DMA。

2.26.206.2 接口定义

函数接口	void LL_USART_EnableDMADeactOnRxErr (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.207 LL_USART_DisableDMADeactOnRxErr

2.26.207.1 功能介绍

接收出错时不禁止 DMA。

2.26.207.2 接口定义

函数接口	void LL_USART_DisableDMADeactOnRxErr (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.208 LL_USART_IsEnabledDMADeactOnRxErr

2.26.208.1 功能介绍

检查接收出错时是否禁止 DMA。

2.26.208.2 接口定义

函数接口	uint32_t LL_USART_IsEnabledDMADeactOnRxErr (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.209 LL_USART_DMA_GetRegAddr

2.26.209.1 功能介绍

获取用于 DMA 传输的 USART 数据寄存器地址。

2.26.209.2 接口定义

函数接口	uint32_t LL_USART_DMA_GetRegAddr (USART_TypeDef * USARTx, uint32_t Direction)
输入	USART_TypeDef * USARTx uint32_t Direction: { LL_USART_DMA_REG_DATA_TRANSMIT, LL_USART_DMA_REG_DATA_RECEIVE }
输出	无
返回值	无
资源使用	
说明	

2.26.210 LL_USART_ReceiveData8

2.26.210.1 功能介绍

读取接收数据寄存器(接收数据值, 8 位)。

2.26.210.2 接口定义

函数接口	uint8_t LL_USART_ReceiveData8 (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0x00, 0xFF]
资源使用	
说明	

2.26.211 LL_USART_ReceiveData9

2.26.211.1 功能介绍

读取接收数据寄存器(接收数据值, 9 位)。

2.26.211.2 接口定义

函数接口	uint16_t LL_USART_ReceiveData9 (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	[0x000, 0x1FF]
资源使用	
说明	

2.26.212 LL_USART_TransmitData8

2.26.212.1 功能介绍

写入发送数据寄存器(发送数据值, 8 位)。

2.26.212.2 接口定义

函数接口	void LL_USART_TransmitData8 (USART_TypeDef * USARTx, uint8_t Value)
输入	USART_TypeDef * USARTx uint8_t Value:

	[0x00, 0xFF]
输出	无
返回值	无
资源使用	
说明	

2.26.213 LL_USART_TransmitData9

2.26.213.1 功能介绍

写入发送数据寄存器(发送数据值, 9 位)。

2.26.213.2 接口定义

函数接口	void LL_USART_TransmitData9 (USART_TypeDef * USARTx, uint16_t Value)
输入	USART_TypeDef * USARTx uint16_t Value: [0x000, 0x1FF]
输出	无
返回值	无
资源使用	
说明	

2.26.214 LL_USART_RequestAutoBaudRate

2.26.214.1 功能介绍

请求波特率自适应。

2.26.214.2 接口定义

函数接口	void LL_USART_RequestAutoBaudRate (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.215 LL_USART_RequestBreakSending

2.26.215.1 功能介绍

请求中断发送。

2.26.215.2 接口定义

函数接口	void LL_USART_RequestBreakSending (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	

说明	
----	--

2.26.216 LL_USART_RequestEnterMuteMode

2.26.216.1 功能介绍

将 USART 设置为静音模式，并设置 RWU 标志。

2.26.216.2 接口定义

函数接口	void LL_USART_RequestEnterMuteMode (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.217 LL_USART_RequestRxDataFlush

2.26.217.1 功能介绍

请求接收数据和 FIFO 刷新。

2.26.217.2 接口定义

函数接口	void LL_USART_RequestRxDataFlush (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.218 LL_USART_RequestTxDataFlush

2.26.218.1 功能介绍

请求发送数据和 FIFO 刷新。

2.26.218.2 接口定义

函数接口	void LL_USART_RequestTxDataFlush (USART_TypeDef * USARTx)
输入	USART_TypeDef * USARTx
输出	无
返回值	无
资源使用	
说明	

2.26.219 LL_USART_DeInit

2.26.219.1 功能介绍

清除 USART 初始化参数。

2.26.219.2 接口定义

函数接口	ErrorStatus LL_USART_DeInit (USART_TypeDef * USARTx)
------	--

输入	USART_TypeDef * USARTx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.26.220 LL_USART_Init

2.26.220.1 功能介绍

USART 初始化。

2.26.220.2 接口定义

函数接口	ErrorStatus LL_USART_Init (USART_TypeDef * USARTx, LL_USART_InitTypeDef * USART_InitStruct)
输入	USART_TypeDef * USARTx LL_USART_InitTypeDef * USART_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.26.221 LL_USART_StructInit

2.26.221.1 功能介绍

USART 结构体初始化。

2.26.221.2 接口定义

函数接口	void LL_USART_StructInit (LL_USART_InitTypeDef * USART_InitStruct)
输入	LL_USART_InitTypeDef * USART_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.26.222 LL_USART_ClockInit

2.26.222.1 功能介绍

USART 时钟初始化。

2.26.222.2 接口定义

函数接口	ErrorStatus LL_USART_ClockInit (USART_TypeDef * USARTx, LL_USART_ClockInitTypeDef * USART_ClockInitStruct)
输入	USART_TypeDef * USARTx LL_LPUART_ClockInitTypeDef * USART_ClockInitStruct
输出	无
返回值	ErrorStatus
资源使用	

说明	
----	--

2.26.223 LL_USART_ClockStructInit

2.26.223.1 功能介绍

USART 时钟结构体初始化。

2.26.223.2 接口定义

函数接口	void LL_USART_ClockStructInit (LL_USART_ClockInitTypeDef * USART_ClockInitStruct)
输入	LL_LPUART_ClockInitTypeDef * USART_ClockInitStruct
输出	无
返回值	无
资源使用	
说明	

2.27 UTILS 模块

属性	类型	字段名	含义
LL_UTILS_PLLInitTypeDef			
读写	uint32_t	PLLN	PLL 输入时钟的分频因子 N
读写	uint32_t	PLLM	PLL 输出时钟的倍频因子 M
读写	uint32_t	PLLR	PLL VCO 的 R 分频系数
LL_UTILS_ClkInitTypeDef			
读写	uint32_t	AHBCLKDivider	AHB 时钟分频器
读写	uint32_t	APB1CLKDivider	APB1 时钟分频器
读写	uint32_t	APB2CLKDivider	APB2 时钟分频器

2.27.1 LL_GetUID_Word0

2.27.1.1 功能介绍

获取唯一设备标识符的 Word0。

2.27.1.2 接口定义

函数接口	uint32_t LL_GetUID_Word0 (void)
输入	无
输出	无
返回值	UID [31:0]
资源使用	
说明	

2.27.2 LL_GetUID_Word1

2.27.2.1 功能介绍

获取唯一设备标识符的 Word1。

2.27.2.2 接口定义

函数接口	uint32_t LL_GetUID_Word1 (void)
输入	无
输出	无
返回值	UID [63:32]
资源使用	
说明	

2.27.3 LL_GetUID_Word2

2.27.3.1 功能介绍

获取唯一设备标识符的 Word2。

2.27.3.2 接口定义

函数接口	uint32_t LL_GetUID_Word2 (void)
输入	无
输出	无
返回值	UID [95:64]
资源使用	
说明	

2.27.4 LL_GetFlashSize

2.27.4.1 功能介绍

获取 FLASH 内存大小。

2.27.4.2 接口定义

函数接口	uint32_t LL_GetFlashSize (void)
输入	无
输出	无
返回值	{ LL_UTILS_USERFLASH_16, LL_UTILS_USERFLASH_32, LL_UTILS_USERFLASH_64, LL_UTILS_USERFLASH_128, LL_UTILS_USERFLASH_236 }
资源使用	
说明	

2.27.5 LL_GetSRAMSize

2.27.5.1 功能介绍

获取 SRAM 内存大小。

2.27.5.2 接口定义

函数接口	uint32_t LL_GetSRAMSize (void)
输入	无

输出	无
返回值	{ LL_UTILS_SRAMSIZE_16, LL_UTILS_SRAMSIZE_32, LL_UTILS_SRAMSIZE_64, LL_UTILS_SRAMSIZE_128, LL_UTILS_SRAMSIZE_236 }
资源使用	
说明	

2.27.6 LL_GetPackageType

2.27.6.1 功能介绍

获取封装类型。

2.27.6.2 接口定义

函数接口	uint32_t LL_GetPackageType (void)
输入	无
输出	无
返回值	{ LL_UTILS_PACKAGETYPE_LQFP48, LL_UTILS_PACKAGETYPE_LQFP44, LL_UTILS_PACKAGETYPE_LQFP32, LL_UTILS_PACKAGETYPE_LQFP28, LL_UTILS_PACKAGETYPE_LQFP24 }
资源使用	
说明	

2.27.7 LL_GetDeviceType

2.27.7.1 功能介绍

获取设备类型。

2.27.7.2 接口定义

函数接口	uint32_t LL_GetDeviceType (void)
输入	无
输出	无
返回值	{ LL_UTILS_DEVICETYPE_TX3602 }
资源使用	
说明	

2.27.8 LL_InitTick

2.27.8.1 功能介绍

配置时间基准的 Cortex-M SysTick 源。

2.27.8.2 接口定义

函数接口	void LL_InitTick (uint32_t HCLKFrequency, uint32_t Ticks)
输入	uint32_t HCLKFrequency uint32_t Ticks
输出	无
返回值	无

资源使用	
说明	

2.27.9 LL_Init1msTick

2.27.9.1 功能介绍

配置 Cortex-M+ SysTick 源的时间基数为 1ms。

2.27.9.2 接口定义

函数接口	void LL_Init1msTick (uint32_t HCLKFrequency)
输入	uint32_t HCLKFrequency
输出	无
返回值	无
资源使用	
说明	

2.27.10 LL_mDelay

2.27.10.1 功能介绍

基于 SysTick 计数器标志的精确延迟(毫秒)。

2.27.10.2 接口定义

函数接口	void LL_mDelay (uint32_t Delay)
输入	uint32_t Delay
输出	无
返回值	无
资源使用	
说明	

2.27.11 LL_SetSystemCoreClock

2.27.11.1 功能介绍

直接设置 SystemCoreClock CMSIS 变量。

2.27.11.2 接口定义

函数接口	void LL_SetSystemCoreClock (uint32_t HCLKFrequency)
输入	uint32_t Delay
输出	无
返回值	无
资源使用	
说明	

2.27.12 LL_PLL_ConfigSystemClock_HSI

2.27.12.1 功能介绍

以最大频率配置系统时钟，HSI 作为锁相环的时钟源。

2.27.12.2 接口定义

函数接口	ErrorStatus LL_PLL_ConfigSystemClock_HSI
------	--

	(LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct, LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct)
输入	LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct
输出	无
返回值	无
资源使用	
说明	

2.27.13 LL_PLL_ConfigSystemClock_HSE

2.27.13.1 功能介绍

以最大频率配置系统时钟，HSE 作为锁相环的时钟源。

2.27.13.2 接口定义

函数接口	ErrorStatus LL_PLL_ConfigSystemClock_HSE (uint32_t HSEFrequency, LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct, LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct)
输入	uint32_t HSEFrequency LL_UTILS_PLLInitTypeDef * UTILS_PLLInitStruct LL_UTILS_ClkInitTypeDef * UTILS_ClkInitStruct
输出	无
返回值	无
资源使用	
说明	

2.27.14 LL_SetFlashLatency

2.27.14.1 功能介绍

根据新的频率和电流电压范围更新 Flash 等待状态数。

2.27.14.2 接口定义

函数接口	ErrorStatus LL_SetFlashLatency (uint32_t HCLKFrequency)
输入	uint32_t HCLKFrequency
输出	无
返回值	无
资源使用	
说明	

2.28 WWDG 模块

属性	类型	字段名	含义
WWDG_TypeDef			
读写	uint32_t	CR	控制寄存器
读写	uint32_t	CFR	配置寄存器
读写	uint32_t	SR	状态寄存器

2.28.1 LL_WWDG_Enable

2.28.1.1 功能介绍

使能 WWDG。

2.28.1.2 接口定义

函数接口	void LL_WWDG_Enable (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	无
资源使用	
说明	

2.28.2 LL_WWDG_IsEnabled

2.28.2.1 功能介绍

检查是否使能 WWDG。

2.28.2.2 接口定义

函数接口	uint32_t LL_WWDG_IsEnabled (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	{0,1}
资源使用	
说明	

2.28.3 LL_WWDG_SetCounter

2.28.3.1 功能介绍

设置 WWDG 计数值。

2.28.3.2 接口定义

函数接口	void LL_WWDG_SetCounter (WWDG_TypeDef * WWDGx, uint32_t Counter)
输入	WWDG_TypeDef * WWDGx uint32_t Counter: [0x00, 0x7F]
输出	无
返回值	无
资源使用	
说明	

2.28.4 LL_WWDG_GetCounter

2.28.4.1 功能介绍

获取 WWDG 计数值。

2.28.4.2 接口定义

函数接口	uint32_t LL_WWDG_GetCounter (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	[0x00, 0x7F]
资源使用	
说明	

2.28.5 LL_WWDG_SetPrescaler

2.28.5.1 功能介绍

设置 WWDG 分频值。

2.28.5.2 接口定义

函数接口	void LL_WWDG_SetPrescaler (WWDG_TypeDef * WWDGx, uint32_t Prescaler)
输入	WWDG_TypeDef * WWDGx uint32_t Prescaler: { LL_WWDG_PRESCALER_1, LL_WWDG_PRESCALER_2, LL_WWDG_PRESCALER_4, LL_WWDG_PRESCALER_8, LL_WWDG_PRESCALER_16, LL_WWDG_PRESCALER_32, LL_WWDG_PRESCALER_64, LL_WWDG_PRESCALER_128 }
输出	无
返回值	无
资源使用	
说明	

2.28.6 LL_WWDG_GetPrescaler

2.28.6.1 功能介绍

获取 WWDG 分频值。

2.28.6.2 接口定义

函数接口	uint32_t LL_WWDG_GetPrescaler (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	{ LL_WWDG_PRESCALER_1, LL_WWDG_PRESCALER_2, LL_WWDG_PRESCALER_4, LL_WWDG_PRESCALER_8, LL_WWDG_PRESCALER_16, LL_WWDG_PRESCALER_32, LL_WWDG_PRESCALER_64, LL_WWDG_PRESCALER_128 }
资源使用	
说明	

2.28.7 LL_WWDG_SetWindow

2.28.7.1 功能介绍

设置 WWDG 窗口值。

2.28.7.2 接口定义

函数接口	void LL_WWDG_SetWindow (WWDG_TypeDef * WWDGx, uint32_t Window)
输入	WWDG_TypeDef * WWDGx uint32_t Window: [0x00, 0x7F]
输出	无
返回值	无
资源使用	
说明	

2.28.8 LL_WWDG_GetWindow

2.28.8.1 功能介绍

获取 WWDG 窗口值。

2.28.8.2 接口定义

函数接口	uint32_t LL_WWDG_GetWindow (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	[0x00, 0x7F]
资源使用	
说明	

2.28.9 LL_WWDG_IsActiveFlag_EWKUP

2.28.9.1 功能介绍

检查提前唤醒中断标志。

2.28.9.2 接口定义

函数接口	uint32_t LL_WWDG_IsActiveFlag_EWKUP (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	{0,1}
资源使用	
说明	

2.28.10 LL_WWDG_ClearFlag_EWKUP

2.28.10.1 功能介绍

清除提前唤醒中断标志。

2.28.10.2 接口定义

函数接口	void LL_WWDG_ClearFlag_EWKUP (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无

返回值	无
资源使用	
说明	

2.28.11 LL_WWDG_EnableIT_EWKUP

2.28.11.1 功能介绍

使能 WWDG 提前唤醒中断。

2.28.11.2 接口定义

函数接口	void LL_WWDG_EnableIT_EWKUP (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	无
资源使用	
说明	

2.28.12 LL_WWDG_IsEnabledIT_EWKUP

2.28.12.1 功能介绍

检查是否使能 WWDG 提前唤醒中断。

2.28.12.2 接口定义

函数接口	uint32_t LL_WWDG_IsEnabledIT_EWKUP (WWDG_TypeDef * WWDGx)
输入	WWDG_TypeDef * WWDGx
输出	无
返回值	{0,1}
资源使用	
说明	

2.29 BEMF 模块

属性	类型	字段名	含义
BEMF_TypeDef			
读写	uint32_t	CR	控制寄存器
LL_BEMF_InitTypeDef			
读写	uint32_t	USEL	BEMF U 管脚选择
读写	uint32_t	VSEL	BEMF V 管脚选择
读写	uint32_t	WSEL	BEMF W 管脚选择
读写	uint32_t	MIDSEL	BEMF 输出管脚选择

2.29.1 LL_BEMF_SetUSEL

2.29.1.1 功能介绍

设置 BEMF U 管脚选择。

2.29.1.2 接口定义

函数接口	void LL_BEMF_SetUSEL(BEMF_TypeDef *BEMFcx, uint32_t USEL)
输入	BEMF_TypeDef *BEMFcx uint32_t USEL: { LL_BEMF_USEL_PA3, LL_BEMF_USEL_PB4, LL_BEMF_USEL_NONE }
输出	无
返回值	无
资源使用	
说明	

2.29.2 LL_BEMF_GetUSEL

2.29.2.1 功能介绍

获取 BEMF U 管脚选择。

2.29.2.2 接口定义

函数接口	uint32_t LL_BEMF_GetUSEL(BEMF_TypeDef *BEMFcx)
输入	BEMF_TypeDef *BEMFcx
输出	无
返回值	{ LL_BEMF_USEL_PA3, LL_BEMF_USEL_PB4, LL_BEMF_USEL_NONE }
资源使用	
说明	

2.29.3 LL_BEMF_SetVSEL

2.29.3.1 功能介绍

设置 BEMF V 管脚选择。

2.29.3.2 接口定义

函数接口	void LL_BEMF_SetVSEL(BEMF_TypeDef *BEMFcx, uint32_t VSEL)
输入	BEMF_TypeDef *BEMFcx uint32_t VSEL: { LL_BEMF_VSEL_PA0, LL_BEMF_VSEL_PB5, LL_BEMF_VSEL_NONE }
输出	无
返回值	无
资源使用	
说明	

2.29.4 LL_BEMF_GetVSEL

2.29.4.1 功能介绍

获取 BEMF V 管脚选择。

2.29.4.2 接口定义

函数接口	uint32_t LL_BEMF_GetVSEL(BEMF_TypeDef *BEMFx)
输入	BEMF_TypeDef *BEMFx
输出	无
返回值	{ LL_BEMF_VSEL_PA0 , LL_BEMF_VSEL_PB5 , LL_BEMF_VSEL_NONE }
资源使用	
说明	

2.29.5 LL_BEMF_SetWSEL

2.29.5.1 功能介绍

设置 BEMF W 管脚选择。

2.29.5.2 接口定义

函数接口	void LL_BEMF_SetWSEL(BEMF_TypeDef *BEMFx, uint32_t WSEL)
输入	BEMF_TypeDef *BEMFx uint32_t WSEL: { LL_BEMF_WSEL_PA1 , LL_BEMF_WSEL_PB6 , LL_BEMF_WSEL_NONE }
输出	无
返回值	无
资源使用	
说明	

2.29.6 LL_BEMF_GetWSEL

2.29.6.1 功能介绍

获取 BEMF W 管脚选择。

2.29.6.2 接口定义

函数接口	uint32_t LL_BEMF_GetWSEL(BEMF_TypeDef *BEMFx)
输入	BEMF_TypeDef *BEMFx
输出	无
返回值	{ LL_BEMF_WSEL_PA1 , LL_BEMF_WSEL_PB6 , LL_BEMF_WSEL_NONE }
资源使用	
说明	

2.29.7 LL_BEMF_SetMIDSEL

2.29.7.1 功能介绍

设置 BEMF 输出管脚选择。

2.29.7.2 接口定义

函数接口	void LL_BEMF_SetMIDSEL(BEMF_TypeDef *BEMFx, uint32_t MIDSel)
输入	BEMF_TypeDef *BEMFx uint32_t MIDSel: { LL_BEMF_MIDSEL_PA2, LL_BEMF_MIDSEL_PB7, LL_BEMF_MIDSEL_NONE }
输出	无
返回值	无
资源使用	
说明	

2.29.8 LL_BEMF_GetMIDSEL

2.29.8.1 功能介绍

获取 BEMF 输出管脚选择。

2.29.8.2 接口定义

函数接口	uint32_t LL_BEMF_GetMIDSEL(BEMF_TypeDef *BEMFx)
输入	BEMF_TypeDef *BEMFx
输出	无
返回值	{ LL_BEMF_MIDSEL_PA2, LL_BEMF_MIDSEL_PB7, LL_BEMF_MIDSEL_NONE }
资源使用	
说明	

2.29.9 LL_BEMF_DeInit

2.29.9.1 功能介绍

清除 BEMF 初始化配置。

2.29.9.2 接口定义

函数接口	ErrorStatus LL_BEMF_DeInit(BEMF_TypeDef *BEMFx)
输入	BEMF_TypeDef *BEMFx
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.29.10 LL_BEMF_Init

2.29.10.1 功能介绍

BEMF 初始化。

2.29.10.2 接口定义

函数接口	ErrorStatus LL_BEMF_Init(BEMF_TypeDef *BEMFx, LL_BEMF_InitTypeDef *BEMF_InitStruct)
输入	BEMF_TypeDef *BEMFx

	LL_BEMF_InitTypeDef *BEMF_InitStruct
输出	无
返回值	ErrorStatus
资源使用	
说明	

2.29.11 LL_BEMF_StructInit

2.29.11.1 功能介绍

BEMF 结构体初始化。

2.29.11.2 接口定义

函数接口	void LL_BEMF_StructInit(LL_BEMF_InitTypeDef *BEMF_InitStruct)
输入	LL_BEMF_InitTypeDef *BEMF_InitStruct
输出	无
返回值	无
资源使用	
说明	

2.30 CORDIC 模块

属性	类型	字段名	含义
CORDIC_TypeDef			
读写	uint32_t	CSR	控制寄存器
读写	uint32_t	WDATA	参数寄存器
读写	uint32_t	RDATA	结果寄存器

2.30.1 LL_CORDIC_Config

2.30.1.1 功能介绍

设置 Cordic 参数。

2.30.1.2 接口定义

函数接口	void LL_CORDIC_Config(CORDIC_TypeDef *CORDICx, uint32_t Function, uint32_t Precision, uint32_t NbWrite, uint32_t NbRead, uint32_t InSize, uint32_t OutSize)
输入	CORDIC_TypeDef *CORDICx uint32_t Function: { LL_CORDIC_FUNCTION_COSINE , LL_CORDIC_FUNCTION_SINE , LL_CORDIC_FUNCTION_PHASE , LL_CORDIC_FUNCTION_MODULUS } uint32_t Precision: { LL_CORDIC_PRECISION_1CYCLE , LL_CORDIC_PRECISION_1CYCLE , LL_CORDIC_PRECISION_2CYCLE , LL_CORDIC_PRECISION_3CYCLE , LL_CORDIC_PRECISION_4CYCLE , LL_CORDIC_PRECISION_5CYCLE ,

	<pre> LL_CORDIC_PRECISION_6CYCLE , LL_CORDIC_PRECISION_7CYCLE , LL_CORDIC_PRECISION_8CYCLE , LL_CORDIC_PRECISION_9CYCLE , LL_CORDIC_PRECISION_10CYCLE , LL_CORDIC_PRECISION_11CYCLE , LL_CORDIC_PRECISION_12CYCLE , LL_CORDIC_PRECISION_13CYCLE , LL_CORDIC_PRECISION_14CYCLE , LL_CORDIC_PRECISION_15CYCLE } uint32_t NbWrite: { LL_CORDIC_NBWRITE_1 , LL_CORDIC_NBWRITE_2 } uint32_t NbRead: { LL_CORDIC_NBREAD_1 , LL_CORDIC_NBREAD_2 } uint32_t InSize: { LL_CORDIC_INSIZE_32BITS , LL_CORDIC_INSIZE_16BITS } uint32_t OutSize: { LL_CORDIC_OUTSIZE_32BITS , LL_CORDIC_OUTSIZE_16BITS } </pre>
输出	无
返回值	无
资源使用	
说明	

2.30.2 LL_CORDIC_SetFunction

2.30.2.1 功能介绍

设置 Cordic 功能。

2.30.2.2 接口定义

函数接口	void LL_CORDIC_SetFunction(CORDIC_TypeDef *CORDICx, uint32_t Function)
输入	<pre> CORDIC_TypeDef *CORDICx uint32_t Function: { LL_CORDIC_FUNCTION_COSINE LL_CORDIC_FUNCTION_SINE , LL_CORDIC_FUNCTION_PHASE , LL_CORDIC_FUNCTION_MODULUS } </pre>
输出	无
返回值	无
资源使用	
说明	

2.30.3 LL_CORDIC_GetFunction

2.30.3.1 功能介绍

获取 Cordic 功能。

2.30.3.2 接口定义

函数接口	uint32_t LL_CORDIC_GetFunction(CORDIC_TypeDef *CORDICx)
------	---

输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{ LL_CORDIC_FUNCTION_COSINE LL_CORDIC_FUNCTION_SINE , LL_CORDIC_FUNCTION_PHASE , LL_CORDIC_FUNCTION_MODULUS }
资源使用	
说明	

2.30.4 LL_CORDIC_SetPrecision

2.30.4.1 功能介绍

设置 Cordic 迭代次数。

2.30.4.2 接口定义

函数接口	void LL_CORDIC_SetPrecision(CORDIC_TypeDef *CORDICx, uint32_t Precision)
输入	CORDIC_TypeDef *CORDICx uint32_t Precision: { LL_CORDIC_PRECISION_1CYCLE , LL_CORDIC_PRECISION_1CYCLE , LL_CORDIC_PRECISION_2CYCLE , LL_CORDIC_PRECISION_3CYCLE , LL_CORDIC_PRECISION_4CYCLE , LL_CORDIC_PRECISION_5CYCLE , LL_CORDIC_PRECISION_6CYCLE , LL_CORDIC_PRECISION_7CYCLE , LL_CORDIC_PRECISION_8CYCLE , LL_CORDIC_PRECISION_9CYCLE , LL_CORDIC_PRECISION_10CYCLE , LL_CORDIC_PRECISION_11CYCLE , LL_CORDIC_PRECISION_12CYCLE , LL_CORDIC_PRECISION_13CYCLE , LL_CORDIC_PRECISION_14CYCLE , LL_CORDIC_PRECISION_15CYCLE }
输出	无
返回值	无
资源使用	
说明	

2.30.5 LL_CORDIC_GetPrecision

2.30.5.1 功能介绍

获取 Cordic 迭代次数。

2.30.5.2 接口定义

函数接口	uint32_t LL_CORDIC_GetPrecision(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无

返回值	{ LL_CORDIC_PRECISION_1CYCLE , LL_CORDIC_PRECISION_1CYCLE , LL_CORDIC_PRECISION_2CYCLE , LL_CORDIC_PRECISION_3CYCLE , LL_CORDIC_PRECISION_4CYCLE , LL_CORDIC_PRECISION_5CYCLE , LL_CORDIC_PRECISION_6CYCLE , LL_CORDIC_PRECISION_7CYCLE , LL_CORDIC_PRECISION_8CYCLE , LL_CORDIC_PRECISION_9CYCLE , LL_CORDIC_PRECISION_10CYCLE , LL_CORDIC_PRECISION_11CYCLE , LL_CORDIC_PRECISION_12CYCLE , LL_CORDIC_PRECISION_13CYCLE , LL_CORDIC_PRECISION_14CYCLE , LL_CORDIC_PRECISION_15CYCLE }
资源使用	
说明	

2.30.6 LL_CORDIC_SetNbWrite

2.30.6.1 功能介绍

设置 Cordic 参数数量。

2.30.6.2 接口定义

函数接口	void LL_CORDIC_SetNbWrite(CORDIC_TypeDef *CORDICx, uint32_t NbWrite)
输入	CORDIC_TypeDef *CORDICx uint32_t NbWrite: { LL_CORDIC_NBWRITE_1 , LL_CORDIC_NBWRITE_2 }
输出	无
返回值	无
资源使用	
说明	

2.30.7 LL_CORDIC_GetNbWrite

2.30.7.1 功能介绍

获取 Cordic 参数数量。

2.30.7.2 接口定义

函数接口	uint32_t LL_CORDIC_GetNbWrite(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{ LL_CORDIC_NBWRITE_1 , LL_CORDIC_NBWRITE_2 }
资源使用	
说明	

2.30.8 LL_CORDIC_SetNbRead

2.30.8.1 功能介绍

设置 Cordic 结果数。

2.30.8.2 接口定义

函数接口	void LL_CORDIC_SetNbRead(CORDIC_TypeDef *CORDICx, uint32_t NbRead)
输入	CORDIC_TypeDef *CORDICx uint32_t NbRead: { LL_CORDIC_NBREAD_1 , LL_CORDIC_NBREAD_2 }
输出	无
返回值	无
资源使用	
说明	

2.30.9 LL_CORDIC_GetNbRead

2.30.9.1 功能介绍

获取 Cordic 结果数。

2.30.9.2 接口定义

函数接口	uint32_t LL_CORDIC_GetNbRead(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{ LL_CORDIC_NBREAD_1 , LL_CORDIC_NBREAD_2 }
资源使用	
说明	

2.30.10 LL_CORDIC_SetInSize

2.30.10.1 功能介绍

设置 Cordic 输入数据宽度。

2.30.10.2 接口定义

函数接口	void LL_CORDIC_SetInSize(CORDIC_TypeDef *CORDICx, uint32_t InSize)
输入	CORDIC_TypeDef *CORDICx uint32_t InSize: { LL_CORDIC_INSIZE_32BITS , LL_CORDIC_INSIZE_16BITS }
输出	无
返回值	无
资源使用	
说明	

2.30.11 LL_CORDIC_GetInSize

2.30.11.1 功能介绍

获取 Cordic 输入数据宽度。

2.30.11.2 接口定义

函数接口	uint32_t LL_CORDIC_GetInSize(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{ LL_CORDIC_INSIZE_32BITS , LL_CORDIC_INSIZE_16BITS }
资源使用	
说明	

2.30.12 LL_CORDIC_SetOutSize

2.30.12.1 功能介绍

设置 Cordic 输出数据宽度。

2.30.12.2 接口定义

函数接口	void LL_CORDIC_SetOutSize(CORDIC_TypeDef *CORDICx, uint32_t OutSize)
输入	CORDIC_TypeDef *CORDICx uint32_t OutSize: { LL_CORDIC_OUTSIZE_32BITS , LL_CORDIC_OUTSIZE_16BITS }
输出	无
返回值	无
资源使用	
说明	

2.30.13 LL_CORDIC_GetOutSize

2.30.13.1 功能介绍

获取 Cordic 输出数据宽度。

2.30.13.2 接口定义

函数接口	uint32_t LL_CORDIC_GetOutSize(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{ LL_CORDIC_OUTSIZE_32BITS , LL_CORDIC_OUTSIZE_16BITS }
资源使用	
说明	

2.30.14 LL_CORDIC_EnableIT

2.30.14.1 功能介绍

使能 CORDIC 中断。

2.30.14.2 接口定义

函数接口	void LL_CORDIC_EnableIT(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	无

资源使用	
说明	

2.30.15LL_CORDIC_DisableIT

2.30.15.1 功能介绍

禁止使能 CORDIC 中断。

2.30.15.2 接口定义

函数接口	void LL_CORDIC_DisableIT(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	无
资源使用	
说明	

2.30.16LL_CORDIC_IsEnabledIT

2.30.16.1 功能介绍

检查是否使能 CORDIC 中断。

2.30.16.2 接口定义

函数接口	uint32_t LL_CORDIC_IsEnabledIT(CORDIC_TypeDef *CORDICx)
输入	无
输出	无
返回值	{0,1}
资源使用	
说明	

2.30.17LL_CORDIC_EnableDMAReq_RD

2.30.17.1 功能介绍

使能 CORDIC DMA 读请求。

2.30.17.2 接口定义

函数接口	void LL_CORDIC_EnableDMAReq_RD(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	无
资源使用	
说明	

2.30.18LL_CORDIC_DisableDMAReq_RD

2.30.18.1 功能介绍

关闭 CORDIC DMA 读请求。

2.30.18.2 接口定义

函数接口	void LL_CORDIC_DisableDMAReq_RD(CORDIC_TypeDef *CORDICx)
------	--

输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	无
资源使用	
说明	

2.30.19LL_CORDIC_IsEnabledDMAReq_RD

2.30.19.1 功能介绍

检查是否使能 CORDIC DMA 读请求。

2.30.19.2 接口定义

函数接口	uint32_t LL_CORDIC_IsEnabledDMAReq_RD(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{0,1}
资源使用	
说明	

2.30.20LL_CORDIC_EnableDMAReq_WR

2.30.20.1 功能介绍

使能 CORDIC DMA 写请求。

2.30.20.2 接口定义

函数接口	void LL_CORDIC_EnableDMAReq_WR(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	无
资源使用	
说明	

2.30.21LL_CORDIC_DisableDMAReq_WR

2.30.21.1 功能介绍

关闭 CORDIC DMA 写请求。

2.30.21.2 接口定义

函数接口	void LL_CORDIC_DisableDMAReq_WR(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	无
资源使用	
说明	

2.30.22 LL_CORDIC_IsEnabledDMAReq_WR

2.30.22.1 功能介绍

检查是否使能 CORDIC DMA 写请求。

2.30.22.2 接口定义

函数接口	uint32_t LL_CORDIC_IsEnabledDMAReq_WR(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{0,1}
资源使用	
说明	

2.30.23 LL_CORDIC_DMA_GetRegAddr

2.30.23.1 功能介绍

获取 Cordic 用于 DMA 传输的寄存器地址。

2.30.23.2 接口定义

函数接口	uint32_t LL_CORDIC_DMA_GetRegAddr(CORDIC_TypeDef *CORDICx, uint32_t Direction)
输入	CORDIC_TypeDef *CORDICx uint32_t Direction: { LL_CORDIC_DMA_REG_DATA_IN, LL_CORDIC_DMA_REG_DATA_OUT }
输出	无
返回值	寄存器地址值
资源使用	
说明	

2.30.24 LL_CORDIC_IsActiveFlag_RRDY

2.30.24.1 功能介绍

检查 CORDIC 计算完成标志状态。

2.30.24.2 接口定义

函数接口	uint32_t LL_CORDIC_IsActiveFlag_RRDY(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	{0,1}
资源使用	
说明	

2.30.25 LL_CORDIC_WriteData

2.30.25.1 功能介绍

写 CORDIC 参数值。

2.30.25.2 接口定义

函数接口	void LL_CORDIC_WriteData(CORDIC_TypeDef *CORDICx, uint32_t InData)
输入	CORDIC_TypeDef *CORDICx uint32_t InData: [0,0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

2.30.26LL_CORDIC_ReadData

2.30.26.1 功能介绍

读 CORDIC 参数值。

2.30.26.2 接口定义

函数接口	uint32_t LL_CORDIC_ReadData(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	[0,0xFFFFFFFF]
资源使用	
说明	

2.30.27LL_CORDIC_DeInit

2.30.27.1 功能介绍

清除 CORDIC 初始化配置。

2.30.27.2 接口定义

函数接口	ErrorStatus LL_CORDIC_DeInit(CORDIC_TypeDef *CORDICx)
输入	CORDIC_TypeDef *CORDICx
输出	无
返回值	ErrorStatus
资源使用	
说明	

3 API（基础 HAL 层）

宏定义	数值	含义
HAL_StatusTypeDef		
HAL_OK	0x00	正确
HAL_ERROR	0x01	错误
HAL_BUSY	0x02	忙
HAL_TIMEOUT	0x03	超时
HAL_LockTypeDef		
HAL_UNLOCKED	0x00	解锁
HAL_LOCKED	0x01	锁定
FunctionalState		
DISABLE	DISABLE	DISABLE
ENABLE	ENABLE	ENABLE

3.1 IWDG 模块

属性	类型	字段名	含义
IWDG_TypeDef			
只写	uint32_t	KR	控制寄存器
读写	uint32_t	PR	预分频寄存器
读写	uint32_t	RLR	重载寄存器
只读	uint32_t	SR	状态寄存器
读写	uint32_t	WINR	窗口寄存器
IWDG_InitTypeDef			
读写	uint32_t	Prescaler	分频值
读写	uint32_t	Reload	重载值
读写	uint32_t	Window	窗口值
IWDG_HandleTypeDef			
读写	IWDG_TypeDef	Instance	寄存器基础地址
读写	IWDG_InitTypeDef	Init	IWDG 初始化参数

3.1.1 HAL_IWDG_Refresh

3.1.1.1 功能介绍

刷新 IWDG。

3.1.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IWDG_Refresh (IWDG_HandleTypeDef * hiwdg)
输入	IWDG_HandleTypeDef * hiwdg
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.1.2 HAL_IWDG_Init

3.1.2.1 功能介绍

初始化并启动 IWDG。

3.1.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IWDG_Init (IWDG_HandleTypeDef * hiwdg)
输入	IWDG_HandleTypeDef * hiwdg
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.2 CORTEX 模块

宏定义	数值	含义
IRQn_Type		
Reset_IRQn	-15	复位矢量, 在通电和热复位时调用
NonMaskableInt_IRQn	-14	不可屏蔽中断, 不能停止或抢占
HardFault_IRQn	-13	硬故障, 各类故障
SVC_IRQn	-5	通过 SVC 指令进行系统服务调用
PendSV_IRQn	-2	可挂起的系统服务请求
SysTick_IRQn	-1	系统时钟定时器
WWDG_IRQn	0	WWDG 中断
LVD_IRQn	1	LVD 中断
RTC_IRQn	2	RTC 中断
FLASH_IRQn	3	FLASH 中断
RCC_IRQn	4	RCC 中断
EXTI0_1_IRQn	5	EXTI0_1 中断
EXTI2_3_IRQn	6	EXTI2_3 中断
EXTI4_15_IRQn	7	EXTI4_15 中断

PLA_IRQn	8	PLA 中断
DMA1_Channel0_IRQn	9	DMA1_Channel0 中断
DMA1_Channel1_DMA1_Channel2_IRQn	10	DMA1_Channel1_DMA1_Channel2 中断
DMA1_Channel3_DMAMUX_IRQn	11	DMA1_Channel3 中断
COMP_IRQn	12	COMP 中断
TIM1_BRK_UP_TRG_COM_IRQn	13	TIM1 中断、更新、触发
TIM1_CC_IRQn	14	TIM1 捕获比较中断
TIM2_IRQn	15	TIM2 中断
TIM3_IRQn	16	TIM3 中断
LPTIM1_IRQn	17	LPTIM1 中断
TIM4_IRQn	18	TIM4 中断
TIM5_IRQn	19	TIM5 中断
TIM6_DAC_IRQn	20	TIM6_DAC 中断
TIM7_IRQn	21	TIM7 中断
ATK_IRQn	22	ATK 中断
I2C1_IRQn	23	I2C1 中断
I2C2_IRQn	24	I2C2 中断
SPI1_IRQn	25	SPI1 中断
SPI2_IRQn	26	SPI2 中断
USART1_IRQn	27	USART1 中断
USART2_IRQn	28	USART2 中断
LPUART1_IRQn	29	LPUART1 中断
USART3_IRQn	30	USART3 中断
ADC_IRQn	31	ADC 中断

3.2.1 HAL_NVIC_SetPriority

3.2.1.1 功能介绍

设置中断的优先级。

3.2.1.2 接口定义

函数接口	void HAL_NVIC_SetPriority (IRQn_Type IRQn, uint32_t PreemptPriority, uint32_t SubPriority)
输入	IRQn_Type IRQn uint32_t PreemptPriority: [0, 3] uint32_t SubPriority

输出	无
返回值	无
资源使用	
说明	

3.2.2 HAL_NVIC_EnableIRQ

3.2.2.1 功能介绍

在 NVIC 中断控制器中使能特定于设备的中断。

3.2.2.2 接口定义

函数接口	void HAL_NVIC_EnableIRQ (IRQn_Type IRQn)
输入	IRQn_Type IRQn
输出	无
返回值	无
资源使用	
说明	

3.2.3 HAL_NVIC_DisableIRQ

3.2.3.1 功能介绍

在 NVIC 中断控制器中禁止特定于设备的中断。

3.2.3.2 接口定义

函数接口	void HAL_NVIC_DisableIRQ (IRQn_Type IRQn)
输入	IRQn_Type IRQn
输出	无
返回值	无
资源使用	
说明	

3.2.4 HAL_NVIC_SystemReset

3.2.4.1 功能介绍

发起系统复位请求，复位 MCU。

3.2.4.2 接口定义

函数接口	void HAL_NVIC_SystemReset (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.2.5 HAL_SYSTICK_Config

3.2.5.1 功能介绍

发起系统复位请求，复位 MCU。

3.2.5.2 接口定义

函数接口	uint32_t HAL_SYSTICK_Config (uint32_t TicksNumb)
输入	uint32_t TicksNumb
输出	无
返回值	{0, 1}
资源使用	
说明	

3.2.6 HAL_NVIC_GetPriority

3.2.6.1 功能介绍

获取中断的优先级。

3.2.6.2 接口定义

函数接口	uint32_t HAL_NVIC_GetPriority (IRQn_Type IRQn)
输入	IRQn_Type IRQn
输出	无
返回值	无
资源使用	
说明	

3.2.7 HAL_NVIC_GetPendingIRQ

3.2.7.1 功能介绍

获取挂起的中断(读取 NVIC 中的挂起寄存器并返回指定中断的挂起位)。

3.2.7.2 接口定义

函数接口	uint32_t HAL_NVIC_GetPendingIRQ (IRQn_Type IRQn)
输入	IRQn_Type IRQn
输出	无
返回值	{0, 1}
资源使用	
说明	

3.2.8 HAL_NVIC_SetPendingIRQ

3.2.8.1 功能介绍

设置外部中断的挂起位。

3.2.8.2 接口定义

函数接口	void HAL_NVIC_SetPendingIRQ (IRQn_Type IRQn)
输入	IRQn_Type IRQn
输出	无
返回值	无
资源使用	
说明	

3.2.9 HAL_NVIC_ClearPendingIRQ

3.2.9.1 功能介绍

清除外部中断的挂起位。

3.2.9.2 接口定义

函数接口	void HAL_NVIC_ClearPendingIRQ (IRQn_Type IRQn)
输入	IRQn_Type IRQn
输出	无
返回值	无
资源使用	
说明	

3.2.10 HAL_SYSTICK_CLKSourceConfig

3.2.10.1 功能介绍

配置 SysTick 时钟源。

3.2.10.2 接口定义

函数接口	void HAL_SYSTICK_CLKSourceConfig (uint32_t CLKSource)
输入	uint32_t CLKSource: { SYSTICK_CLKSOURCE_HCLK_DIV8 , SYSTICK_CLKSOURCE_HCLK }
输出	无
返回值	无
资源使用	
说明	

3.2.11 HAL_SYSTICK_IRQHandler

3.2.11.1 功能介绍

处理 SYSTICK 中断请求。

3.2.11.2 接口定义

函数接口	void HAL_SYSTICK_IRQHandler (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.2.12 HAL_SYSTICK_Callback

3.2.12.1 功能介绍

SYSTICK 回调。

3.2.12.2 接口定义

函数接口	void HAL_SYSTICK_Callback (void)
------	------------------------------------

输入	无
输出	无
返回值	无
资源使用	
说明	

3.3 EXTI 模块

属性	类型	字段名	含义
EXTI_HandleTypeDef			
读写	uint32_t	Line	EXTI 行号
读写	指针函数	void(*RisingCallback)(void)	上升沿回调
读写	指针函数	void(*FallingCallback)(void)	下降沿回调
EXTI_ConfigTypeDef			
读写	uint32_t	Line	EXTI 行号
读写	uint32_t	Mode	模式
读写	uint32_t	Trigger	触发源
读写	uint32_t	GPIOSel	GPIO 复用选择

3.3.1 HAL_EXTI_SetConfigLine

3.3.1.1 功能介绍

设置 EXTI 的配置

3.3.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_EXTI_SetConfigLine (EXTI_HandleTypeDef * hexiti, EXTI_ConfigTypeDef * pExtiConfig)
输入	EXTI_HandleTypeDef * hexiti EXTI_ConfigTypeDef * pExtiConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.3.2 HAL_EXTI_GetConfigLine

3.3.2.1 功能介绍

获取 EXTI 的配置

3.3.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_EXTI_GetConfigLine (EXTI_HandleTypeDef * hexiti, EXTI_ConfigTypeDef * pExtiConfig)
输入	EXTI_HandleTypeDef * hexiti
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.3.3 HAL_EXTI_ClearConfigLine

3.3.3.1 功能介绍

清除 EXTI 的配置

3.3.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_EXTI_ClearConfigLine (EXTI_HandleTypeDef * hexiti)
输入	EXTI_HandleTypeDef * hexiti
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.3.4 HAL_EXTI_RegisterCallback

3.3.4.1 功能介绍

EXTI 回调。

3.3.4.2 接口定义

函数接口	HAL_StatusTypeDef HAL_EXTI_RegisterCallback (EXTI_HandleTypeDef * hexiti, EXTI_CallbackIDTypeDef CallbackID, void(*)(void) pPendingCbfm)
输入	EXTI_HandleTypeDef * hexiti EXTI_CallbackIDTypeDef CallbackID void(*)(void) pPendingCbfm
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.3.5 HAL_EXTI_GetHandle

3.3.5.1 功能介绍

存储行号作为句柄私有字段。

3.3.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_EXTI_GetHandle (EXTI_HandleTypeDef * hexiti, uint32_t ExtiLine)
输入	EXTI_HandleTypeDef * hexiti uint32_t ExtiLine: { LL_EXTI_LINE_0, LL_EXTI_LINE_1, LL_EXTI_LINE_2, LL_EXTI_LINE_3, LL_EXTI_LINE_4, LL_EXTI_LINE_5, LL_EXTI_LINE_6, LL_EXTI_LINE_7, LL_EXTI_LINE_8, LL_EXTI_LINE_9, LL_EXTI_LINE_10, LL_EXTI_LINE_11,

	LL_EXTI_LINE_12, LL_EXTI_LINE_13, LL_EXTI_LINE_14, LL_EXTI_LINE_15, LL_EXTI_LINE_16, LL_EXTI_LINE_17, LL_EXTI_LINE_18, LL_EXTI_LINE_19, LL_EXTI_LINE_20, LL_EXTI_LINE_21, LL_EXTI_LINE_22, LL_EXTI_LINE_23, LL_EXTI_LINE_24, LL_EXTI_LINE_25, LL_EXTI_LINE_ALL}
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.3.6 HAL_EXTI_IRQHandler

3.3.6.1 功能介绍

处理 EXTI 中断请求。

3.3.6.2 接口定义

函数接口	void HAL_EXTI_IRQHandler (EXTI_HandleTypeDef * hexiti)
输入	EXTI_HandleTypeDef * hexiti
输出	无
返回值	无
资源使用	
说明	

3.3.7 HAL_EXTI_GetPending

3.3.7.1 功能介绍

获取中断挂起位。

3.3.7.2 接口定义

函数接口	uint32_t HAL_EXTI_GetPending (EXTI_HandleTypeDef * hexiti, uint32_t Edge)
输入	EXTI_HandleTypeDef * hexiti uint32_t Edge: { EXTI_TRIGGER_RISING, EXTI_TRIGGER_FALLING }
输出	无
返回值	{0, 1}
资源使用	
说明	

3.3.8 HAL_EXTI_ClearPending

3.3.8.1 功能介绍

清除中断挂起位。

3.3.8.2 接口定义

函数接口	void HAL_EXTI_ClearPending (EXTI_HandleTypeDef * hexiti, uint32_t Edge)
输入	EXTI_HandleTypeDef * hexiti

	uint32_t Edge: { EXTI_TRIGGER_RISING, EXTI_TRIGGER_FALLING }
输出	无
返回值	无
资源使用	
说明	

3.4 FLASH 模块

属性	类型	字段名	含义
FLASH_EraseInitTypeDef			
读写	uint32_t	TypeErase	擦除类型
读写	uint32_t	Page	擦除页数
读写	uint32_t	NbPages	擦除页号
FLASH_OBProgramInitTypeDef			
读写	uint32_t	OptionType	选项字节类型
读写	uint32_t	RDPLLevel	读保护等级
读写	uint32_t	USERType	选项字节用户类型
读写	uint32_t	USERConfig	选项字节用户配置
读写	uint32_t	WRPConfig	写保护区配置
读写	uint32_t	WRP1AStartOffset	写保护区 1A 起始偏移地址
读写	uint32_t	WRP1AEndOffset	写保护区 1A 结束偏移地址
读写	uint32_t	WRP1BStartOffset	写保护区 1B 起始偏移地址
读写	uint32_t	WRP1BEndOffset	写保护区 1B 结束偏移地址
读写	uint32_t	PCROPConfig	代码读出保护区配置
读写	uint32_t	PCROP1AStartOffset	代码读出保护区 1A 起始偏移地址
读写	uint32_t	PCROP1AEndOffset	代码读出保护区 1A 结束偏移地址
读写	uint32_t	PCROP1BStartOffset	代码读出保护区 1B 起始偏移地址
读写	uint32_t	PCROP1BEndOffset	代码读出保护区 1B

			结束偏移地址
读写	uint32_t	BootEntryPoint	启动配置
读写	uint32_t	SecSize	用户安全区域占用 User flash 页的数量

3.4.1 HAL_FLASH_Program

3.4.1.1 功能介绍

在指定地址处编程行的字。

3.4.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_Program (uint32_t Address, uint32_t Data)
输入	uint32_t Address uint32_t Data
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.2 HAL_FLASH_Program_IT

3.4.2.1 功能介绍

在启用中断的情况下，在指定地址处编程行的字。

3.4.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_Program_IT (uint32_t Address, uint32_t Data)
输入	uint32_t Address uint32_t Data
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.3 HAL_FLASH_ProgramHalfWord

3.4.3.1 功能介绍

在指定地址上编程半字。

3.4.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_ProgramHalfWord (uint32_t Address, uint32_t Data)
输入	uint32_t Address uint32_t Data
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.4.4 HAL_FLASH_ProgramHalfWord_IT

3.4.4.1 功能介绍

在启用中断的情况下，在指定地址上编程半字。

3.4.4.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_ProgramHalfWord_IT (uint32_t Address, uint32_t Data)
输入	uint32_t Address uint32_t Data
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.5 HAL_FLASH_IRQHandler

3.4.5.1 功能介绍

处理 FLASH 中断请求。

3.4.5.2 接口定义

函数接口	void HAL_FLASH_IRQHandler (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.4.6 HAL_FLASH_EndOfOperationCallback

3.4.6.1 功能介绍

FLASH 结束操作中断回调。

3.4.6.2 接口定义

函数接口	void HAL_FLASH_EndOfOperationCallback (uint32_t ReturnValue)
输入	uint32_t ReturnValue: 此参数保存的值取决于正在进行的过程 Mass Erase: 0 Page Erase: 已擦除的页面 Program: 已为数据程序选择的地址
输出	无
返回值	无
资源使用	
说明	

3.4.7 HAL_FLASH_OperationErrorCallback

3.4.7.1 功能介绍

FLASH 结束操作中断回调。

3.4.7.2 接口定义

函数接口	void HAL_FLASH_OperationErrorCallback (uint32_t ReturnValue)
输入	uint32_t ReturnValue: 此参数保存的值取决于正在进行的过程 Mass Erase: 0 Page Erase:返回错误的页码 Program:为数据程序选择的地址
输出	无
返回值	无
资源使用	
说明	

3.4.8 HAL_FLASH_Unlock

3.4.8.1 功能介绍

解锁 FLASH 控制寄存器访问。

3.4.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_Unlock (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.9 HAL_FLASH_Lock

3.4.9.1 功能介绍

锁定 FLASH 控制寄存器访问。

3.4.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_Lock (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.10 HAL_FLASH_OB_Unlock

3.4.10.1 功能介绍

解锁 FLASH 选项字节寄存器访问。

3.4.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_OB_Unlock (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.11 HAL_FLASH_OB_Lock

3.4.11.1 功能介绍

锁定 FLASH 选项字节寄存器访问。

3.4.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_OB_Lock (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.12 HAL_FLASH_OB_Launch

3.4.12.1 功能介绍

启动 FLASH 选项字节加载。

3.4.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASH_OB_Launch (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.13 HAL_FLASH_GetError

3.4.13.1 功能介绍

启动 FLASH 选项字节加载。

3.4.13.2 接口定义

函数接口	uint32_t HAL_FLASH_GetError (void)
输入	无
输出	无
返回值	{ HAL_FLASH_ERROR_NONE, HAL_FLASH_ERROR_PROG, HAL_FLASH_ERROR_WRP, HAL_FLASH_ERROR_PGA, HAL_FLASH_ERROR_SIZE, HAL_FLASH_ERROR_PGS, HAL_FLASH_ERROR_RD, HAL_FLASH_ERROR_AC }

资源使用	
说明	

3.4.14 HAL_FLASHEx_Erase

3.4.14.1 功能介绍

执行大量擦除或擦除指定的 FLASH 页。

3.4.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASHEx_Erase (FLASH_EraseInitTypeDef * pEraseInit, uint32_t * PageError)
输入	FLASH_EraseInitTypeDef * pEraseInit uint32_t * PageError
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.15 HAL_FLASHEx_Erase_IT

3.4.15.1 功能介绍

在启用中断的情况下，执行大量擦除或擦除指定的 FLASH 页。

3.4.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASHEx_Erase_IT (FLASH_EraseInitTypeDef * pEraseInit)
输入	FLASH_EraseInitTypeDef * pEraseInit
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.16 HAL_FLASHEx_EnableSecMemProtection

3.4.16.1 功能介绍

启动用户安全区域保护。

3.4.16.2 接口定义

函数接口	void HAL_FLASHEx_EnableSecMemProtection (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.4.17 HAL_FLASHEx_OBProgram

3.4.17.1 功能介绍

编程选项字节。

3.4.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_FLASHEx_OBProgram (FLASH_OBProgramInitTypeDef * pOBInit)
输入	FLASH_OBProgramInitTypeDef * pOBInit
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.4.18 HAL_FLASHEx_OBGetConfig

3.4.18.1 功能介绍

获取选项字节配置。

3.4.18.2 接口定义

函数接口	void HAL_FLASHEx_OBGetConfig (FLASH_OBProgramInitTypeDef * pOBInit)
输入	FLASH_OBProgramInitTypeDef * pOBInit
输出	无
返回值	无
资源使用	
说明	

3.5 GPIO 模块

宏定义	数值	含义
GPIO_PinState		
GPIO_PIN_RESET	0x00	复位
GPIO_PIN_SET	0x01	置位

属性	类型	字段名	含义
GPIO_TypeDef			
读写	uint32_t	MODER	端口模式寄存器
读写	uint32_t	OTYPER	端口输出类型寄存器
读写	uint32_t	PUPDR	端口上拉/ 下拉寄存器
只读	uint32_t	IDR	端口输入寄存器
读写	uint32_t	ODR	端口输出数据寄存器

只写	uint32_t	BSRR	端口置位/ 复位寄存器
读写	uint32_t	LCKR	端口配置锁定寄存器
只写	uint32_t	AFRL	复用功能低位寄存器
只写	uint32_t	AFRH	复用功能高位寄存器
只写	uint32_t	BRR	端口位复位寄存器
GPIO_InitTypeDef			
读写	uint32_t	Pin	需要配置的 GPIO 引脚
读写	uint32_t	Mode	所选引脚的工作模式
读写	uint32_t	Pull	选定引脚的操作上拉/下拉
读写	uint32_t	Alternate	外设要连接到所选引脚

3.5.1 HAL_GPIO_Init

3.5.1.1 功能介绍

GPIO 初始化。

3.5.1.2 接口定义

函数接口	void HAL_GPIO_Init (GPIO_TypeDef * GPIOx, GPIO_InitTypeDef * GPIO_Init)
输入	GPIO_TypeDef * GPIOx GPIO_InitTypeDef * GPIO_Init
输出	无
返回值	无
资源使用	
说明	

3.5.2 HAL_GPIO_DeInit

3.5.2.1 功能介绍

清除 GPIO 初始化配置。

3.5.2.2 接口定义

函数接口	void HAL_GPIO_DeInit (GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)
输入	GPIO_TypeDef * GPIOx

	uint32_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	无
资源使用	
说明	

3.5.3 HAL_GPIO_ReadPin

3.5.3.1 功能介绍

读取指定的 GPIO 输入端口引脚。

3.5.3.2 接口定义

函数接口	GPIO_PinState HAL_GPIO_ReadPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)
输入	GPIO_TypeDef * GPIOx Uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	GPIO_PinState
资源使用	
说明	

3.5.4 HAL_GPIO_WritePin

3.5.4.1 功能介绍

设置指定的 GPIO 输入端口引脚值。

3.5.4.2 接口定义

函数接口	void HAL_GPIO_WritePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin, GPIO_PinState PinState)
输入	GPIO_TypeDef * GPIOx Uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}

	LL_GPIO_PIN_15} GPIO_PinState PinState
输出	无
返回值	无
资源使用	
说明	

3.5.5 HAL_GPIO_TogglePin

3.5.5.1 功能介绍

翻转指定的 GPIO 输入端口引脚值。

3.5.5.2 接口定义

函数接口	void HAL_GPIO_TogglePin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)
输入	GPIO_TypeDef * GPIOx uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	无
资源使用	
说明	

3.5.6 HAL_GPIO_LockPin

3.5.6.1 功能介绍

锁定 GPIO 端口引脚配置。

3.5.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_GPIO_LockPin (GPIO_TypeDef * GPIOx, uint16_t GPIO_Pin)
输入	GPIO_TypeDef * GPIOx uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	无
资源使用	

说明	
----	--

3.5.7 HAL_GPIO_EXTI_IRQHandler

3.5.7.1 功能介绍

处理 EXTI 中断请求。

3.5.7.2 接口定义

函数接口	void HAL_GPIO_EXTI_IRQHandler (uint16_t GPIO_Pin)
输入	Uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	无
资源使用	
说明	

3.5.8 HAL_GPIO_EXTI_Rising_Callback

3.5.8.1 功能介绍

处理 EXTI 上升沿中断请求。

3.5.8.2 接口定义

函数接口	void HAL_GPIO_EXTI_Rising_Callback (uint16_t GPIO_Pin)
输入	Uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2, LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	无
资源使用	
说明	

3.5.9 HAL_GPIO_EXTI_Falling_Callback

3.5.9.1 功能介绍

处理 EXTI 下降沿中断请求。

3.5.9.2 接口定义

函数接口	void HAL_GPIO_EXTI_Falling_Callback (uint16_t GPIO_Pin)
输入	Uint16_t GPIO_Pin: { LL_GPIO_PIN_0, LL_GPIO_PIN_1, LL_GPIO_PIN_2,

	LL_GPIO_PIN_3, LL_GPIO_PIN_4, LL_GPIO_PIN_5, LL_GPIO_PIN_6, LL_GPIO_PIN_7, LL_GPIO_PIN_8, LL_GPIO_PIN_9, LL_GPIO_PIN_10, LL_GPIO_PIN_11, LL_GPIO_PIN_12, LL_GPIO_PIN_13, LL_GPIO_PIN_14, LL_GPIO_PIN_15}
输出	无
返回值	无
资源使用	
说明	

3.6 PWR 模块

属性	类型	字段名	含义
PWR_PVDTypeDef			
读写	uint32_t	Level	PVD 阈值等级
读写	FunctionalState	FilterState	PVD 监测信号数字滤波状态
读写	uint32_t	FilterTime	PVD 监测信号数字滤波时间
只读	uint32_t	Source	PVD 监测信号源
读写	uint32_t	Hysteresis	PVD 迟滞
只写	uint32_t	Mode	模式

3.6.1 HAL_PWR_DeInit

3.6.1.1 功能介绍

清除 PWR 初始化配置。

3.6.1.2 接口定义

函数接口	void HAL_PWR_DeInit (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.2 HAL_PWR_EnterSLEEPMode

3.6.2.1 功能介绍

进入睡眠模式。

3.6.2.2 接口定义

函数接口	void HAL_PWR_EnterSLEEPMode (uint32_t Regulator, uint8_t SLEEPEntry)
------	--

输入	uint32_t Regulator: {PWR_MAINREGULATOR_ON, PWR_LOWPOWERREGULATOR_ON } uint8_t SLEEPEntry: { PWR_SLEEPENTRY_WFI, PWR_SLEEPENTRY_WFE }
输出	无
返回值	无
资源使用	
说明	

3.6.3 HAL_PWR_EnterSTOPMode

3.6.3.1 功能介绍

进入睡眠模式。

3.6.3.2 接口定义

函数接口	void HAL_PWR_EnterSTOPMode (uint32_t Regulator, uint8_t STOPEntry)
输入	uint32_t Regulator: {PWR_MAINREGULATOR_ON, PWR_LOWPOWERREGULATOR_ON } uint8_t STOPEntry: { PWR_STOPENTRY_WFI, PWR_STOPENTRY_WFE }
输出	无
返回值	无
资源使用	
说明	

3.6.4 HAL_PWR_EnableSleepOnExit

3.6.4.1 功能介绍

退出最低优先级中断服务函数时进入低功耗模式。

3.6.4.2 接口定义

函数接口	void HAL_PWR_EnableSleepOnExit (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.5 HAL_PWR_DisableSleepOnExit

3.6.5.1 功能介绍

退出最低优先级中断服务函数时不进入低功耗模式。

3.6.5.2 接口定义

函数接口	void HAL_PWR_DisableSleepOnExit (void)
------	--

输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.6 HAL_PWR_EnableSEVOnPend

3.6.6.1 功能介绍

启用事件和所有中断,包括禁用中断,可以唤醒处理器。

3.6.6.2 接口定义

函数接口	void HAL_PWR_EnableSEVOnPend (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.7 HAL_PWR_DisableSEVOnPend

3.6.7.1 功能介绍

只有启用的中断或事件才能唤醒处理器，禁用的中断将被排除。

3.6.7.2 接口定义

函数接口	void HAL_PWR_DisableSEVOnPend (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.8 HAL_PWR_VREF_VoltageScalingConfig

3.6.8.1 功能介绍

设置电压参考选择。

3.6.8.2 接口定义

函数接口	void HAL_PWR_VREF_VoltageScalingConfig (uint32_t VoltageScaling)
输入	uint32_t VoltageScaling: { PWR_VREF_1V25, PWR_VREF_2V5, PWR_VREF_3V, PWR_VREF_4V }
输出	无
返回值	无
资源使用	
说明	

3.6.9 HAL_PWR_EnableVREF

3.6.9.1 功能介绍

使能 BGR。

3.6.9.2 接口定义

函数接口	void HAL_PWR_EnableVREF (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.10 HAL_PWR_DisableVREF

3.6.10.1 功能介绍

禁止使能 BGR。

3.6.10.2 接口定义

函数接口	void HAL_PWR_DisableVREF (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.11 HAL_PWREx_PVD_Config

3.6.11.1 功能介绍

配置 PVD。

3.6.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_PWREx_PVD_Config (PWR_PVDTypeDef * sConfigPVD)
输入	PWR_PVDTypeDef * sConfigPVD
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.6.12 HAL_PWREx_PVD_Enable

3.6.12.1 功能介绍

使能 PVD。

3.6.12.2 接口定义

函数接口	void HAL_PWREx_PVD_Enable (void)
输入	无

输出	无
返回值	无
资源使用	
说明	

3.6.13 HAL_PWREx_PVD_Disable

3.6.13.1 功能介绍

禁止使能 PVD。

3.6.13.2 接口定义

函数接口	void HAL_PWREx_PVD_Disable (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.14 HAL_PWREx_PVD_IRQHandler

3.6.14.1 功能介绍

处理 PVD 中断请求。

3.6.14.2 接口定义

函数接口	void HAL_PWREx_PVD_IRQHandler (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.15 HAL_PWREx_PVD_Callback

3.6.15.1 功能介绍

PVD 中断回调。

3.6.15.2 接口定义

函数接口	void HAL_PWREx_PVD_Callback (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.16 HAL_PWREx_EnableLowPowerRunMode

3.6.16.1 功能介绍

使能低功耗模式。

3.6.16.2 接口定义

函数接口	void HAL_PWREx_EnableLowPowerRunMode (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.6.17 HAL_PWREx_DisableLowPowerRunMode

3.6.17.1 功能介绍

退出低功耗模式。

3.6.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_PWREx_DisableLowPowerRunMode (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7 RTC 模块

宏定义	数值	含义
HAL_RTCStateTypeDef		
HAL_RTC_STATE_RESET	0x00	重置
HAL_RTC_STATE_READY	0x01	就绪
HAL_RTC_STATE_BUSY	0x02	忙
HAL_RTC_STATE_TIMEOUT	0x03	超时
HAL_RTC_STATE_ERROR	0x04	错误

属性	类型	字段名	含义
RTC_TypeDef			
读写	uint32_t	CR0	控制寄存器 0
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	SEC	秒计数寄存器
读写	uint32_t	MIN	分计数寄存器
读写	uint32_t	HOUR	时计数寄存器
读写	uint32_t	DAY	日计数寄存器
读写	uint32_t	WEEK	周计数寄存器

读写	uint32_t	MON	月计数寄存器
读写	uint32_t	YEAR	年计数寄存器
读写	uint32_t	ALMMIN	分闹钟寄存器
读写	uint32_t	ALMHOUR	时闹钟寄存器
读写	uint32_t	ALMWEEK	周闹钟寄存器
读写	uint32_t	COMPEN	时钟校准寄存器
LL_RTC_InitTypeDef			
读写	uint32_t	HourFormat	RTC 小时格式
读写	uint32_t	PeriodSource	RTC 周期中断源
读写	uint32_t	PITS	RTC 中断周期选择
读写	uint32_t	PITX	设置产生周期中断的时间间隔，可设定的范围为 0.5 秒到 32 秒，步进为 0.5 秒
LL_RTC_TimeTypeDef			
读写	uint32_t	TimeFormat	时间格式
读写	uint8_t	Hours	时
读写	uint8_t	Minutes	分
读写	uint8_t	Seconds	秒
LL_RTC_DateTypeDef			
读写	uint8_t	WeekDay	星期
读写	uint8_t	Month	月份
读写	uint8_t	Day	日
读写	uint8_t	Year	年
LL_RTC_AlarmTypeDef			
读写	uint32_t	TimeFormat	时间格式
读写	uint8_t	AlarmWeek	星期闹钟
读写	uint8_t	AlarmHour	时闹钟
读写	uint8_t	AlarmMinute	分闹钟
RTC_HandleTypeDef			

读写	RTC_TypeDef	Instance	寄存器基础地址
读写	RTC_InitTypeDef	Init	初始化参数
读写	HAL_LockTypeDef	Lock	锁定项
读写	HAL_RTCStateTypeDef	State	状态
读写	指针函数	void(*AlarmEventCallback) (struct__RTC_HandleTypeDef* hrtc)	闹钟事件回调
读写	指针函数	void(*PeriodEventCallback) (struct__RTC_HandleTypeDef* hrtc)	周期事件回调
读写	指针函数	void(*MspInitCallback) (struct__RTC_HandleTypeDef * hrtc)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback) (struct__RTC_HandleTypeDef * hrtc)	MSP 非初始化回调

3.7.1 HAL_RTC_Init

3.7.1.1 功能介绍

RTC 初始化。

3.7.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_Init (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.2 HAL_RTC_DeInit

3.7.2.1 功能介绍

清除 RTC 初始化参数。

3.7.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_DeInit (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.3 HAL_RTC_MspInit

3.7.3.1 功能介绍

RTC MSP 初始化。

3.7.3.2 接口定义

函数接口	void HAL_RTC_MspInit (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	无
资源使用	
说明	

3.7.4 HAL_RTC_MspDeInit

3.7.4.1 功能介绍

清除 RTC MSP 初始化参数。

3.7.4.2 接口定义

函数接口	void HAL_RTC_MspDeInit (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	无
资源使用	
说明	

3.7.5 HAL_RTC_SetTime

3.7.5.1 功能介绍

设置 RTC 当前时间。

3.7.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_SetTime (RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_TimeTypeDef * sTime uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.6 HAL_RTC_GetTime

3.7.6.1 功能介绍

获取 RTC 当前时间。

3.7.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_GetTime (RTC_HandleTypeDef * hrtc, RTC_TimeTypeDef * sTime, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_TimeTypeDef * sTime uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.7 HAL_RTC_SetDate

3.7.7.1 功能介绍

设置 RTC 当前日期。

3.7.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_SetDate (RTC_HandleTypeDef * hrtc, RTC_DateTypeDef * sDate, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_DateTypeDef * sDate uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.8 HAL_RTC_GetDate

3.7.8.1 功能介绍

获取 RTC 当前日期。

3.7.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_GetDate (RTC_HandleTypeDef * hrtc, RTC_DateTypeDef * sDate, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_DateTypeDef * sDate uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.9 HAL_RTC_SetAlarm

3.7.9.1 功能介绍

设置 RTC 闹钟。

3.7.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_SetAlarm (RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_AlarmTypeDef * sAlarm uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.10 HAL_RTC_SetAlarm_IT

3.7.10.1 功能介绍

设置 RTC 闹钟中断。

3.7.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_SetAlarm_IT (RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_AlarmTypeDef * sAlarm uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.11 HAL_RTC_DeactivateAlarm

3.7.11.1 功能介绍

关闭 RTC 闹钟。

3.7.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_DeactivateAlarm (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.12 HAL_RTC_GetAlarm

3.7.12.1 功能介绍

获取 RTC 闹钟。

3.7.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_GetAlarm (RTC_HandleTypeDef * hrtc, RTC_AlarmTypeDef * sAlarm, uint32_t Format)
输入	RTC_HandleTypeDef * hrtc RTC_AlarmTypeDef * sAlarm uint32_t Format: { LL_RTC_FORMAT_BIN, LL_RTC_FORMAT_BCD }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.13 HAL_RTC_SetPeriod_IT

3.7.13.1 功能介绍

设置 RTC 周期中断、RTC 中断率、设置产生周期中断的时间间隔。

3.7.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_SetPeriod_IT (RTC_HandleTypeDef * hrtc, RTC_PeriodTypeDef * sPeriod)
输入	RTC_HandleTypeDef * hrtc RTC_PeriodTypeDef * sPeriod
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.14 HAL_RTC_IRQHandler

3.7.14.1 功能介绍

处理 RTC 中断请求。

3.7.14.2 接口定义

函数接口	void HAL_RTC_IRQHandler (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	无
资源使用	
说明	

3.7.15 HAL_RTC_PollForAlarmEvent

3.7.15.1 功能介绍

处理报警轮询请求。

3.7.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_PollForAlarmEvent (RTC_HandleTypeDef * hrtc, uint32_t Timeout)
输入	RTC_HandleTypeDef * hrtc uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.16 HAL_RTC_AlarmEventCallback

3.7.16.1 功能介绍

RTC 闹钟回调。

3.7.16.2 接口定义

函数接口	void HAL_RTC_AlarmEventCallbac (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	无
资源使用	
说明	

3.7.17 HAL_RTC_PeriodEventCallback

3.7.17.1 功能介绍

RTC 周期回调。

3.7.17.2 接口定义

函数接口	void HAL_RTC_PeriodEventCallback (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	无
资源使用	
说明	

3.7.18 HAL_RTC_SetSmoothCalib

3.7.18.1 功能介绍

设置时钟误差补偿。

3.7.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_SetSmoothCalib (RTC_HandleTypeDef * hrtc, uint32_t SmoothCalibMinusPulsesValue)
输入	RTC_HandleTypeDef * hrtc uint32_t SmoothCalibMinusPulsesValue: [0x00, 0x1FF]
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.19 HAL_RTC_EnterLowpowerMode

3.7.19.1 功能介绍

进入低功耗模式。

3.7.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_EnterLowpowerMode (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.20 HAL_RTC_ExitLowpowerMode

3.7.20.1 功能介绍

退出低功耗模式。

3.7.20.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RTC_ExitLowpowerMode (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.7.21 HAL_RTC_GetState

3.7.21.1 功能介绍

获取 RTC 句柄状态。

3.7.21.2 接口定义

函数接口	HAL_RTCStateTypeDef HAL_RTC_GetState (RTC_HandleTypeDef * hrtc)
输入	RTC_HandleTypeDef * hrtc
输出	无
返回值	HAL_RTCStateTypeDef
资源使用	
说明	

3.8 ADC 模块

属性	类型	字段名	含义
ADC_TypeDef			
读写	uint32_t	CR	控制寄存器
读写	uint32_t	CFG1	配置寄存器 1
读写	uint32_t	CFG2	配置寄存器 2
读写	uint32_t	ISR	中断和状态寄存器
读写	uint32_t	IER	中断使能寄存器
读写	uint32_t	SAMR	采样时间寄存器
读写	uint32_t	CHSELR1	通道选择器寄存器 1
读写	uint32_t	CHSELR2	通道选择器寄存器 2
读写	uint32_t	AWD1TR	看门狗 1 监控电压阈值寄存器
读写	uint32_t	CALFACT	校准系数
只读	uint32_t	DR	数据寄存器
ADC_InitTypeDef			
读写	uint32_t	ClockPrescaler	时钟源分频
读写	uint32_t	DataAlign	数据对齐
读写	uint32_t	Vref	参考电压
读写	uint32_t	EOCSelection	EOC 选择
读写	uint32_t	LowPowerAutoWait	低功耗自动等待
读写	uint32_t	ConvMode	转换模式
读写	uint32_t	NbrOfConversion	转换序列号
读写	uint32_t	ExternalTrigConv	外部触发
读写	uint32_t	ExternalTrigConvEdge	外部触发极性
读写	uint32_t	Overrun	溢出
读写	uint32_t	SamplingTime	采样时间
读写	FunctionalState	ChBufEnable	通道输入 BUFFER 使能
ADC_ChannelConfTypeDef			
读写	uint32_t	Channel	通道

读写	uint32_t	Rank	序列
ADC_AnalogWDGConfTypeDef			
读写	uint32_t	WatchdogNumber	看门狗号
读写	FunctionalState	ITMode	中断模式
读写	uint32_t	HighThreshold	上阈值
读写	uint32_t	LowThreshold	下阈值
ADC_HandleTypeDef			
读写	ADC_TypeDef	Instance	寄存器基础地址
读写	ADC_InitTypeDef	Init	初始化参数
读写	DMA_HandleTypeDef	DMA_Handle	DMA 句柄
读写	HAL_LockTypeDef	Lock	锁定项
读写	uint32_t	State	状态
读写	uint32_t	ErrorCode	错误代码
读写	指针函数	ConvCpltCallback	传输完成回调
读写	指针函数	ConvHalfCpltCallback	传输完成一半回调
读写	指针函数	LevelOutOfWindowCallback	模拟看门狗 1 回调
读写	指针函数	ErrorCallback	错误回调
读写	指针函数	MspInitCallback	MSP 初始化回调
读写	指针函数	MspDeInitCallback	MSP 非初始化回调
ADC_CalibrationConfTypeDef			
读写	uint32_t	GainK	校准参数 K 值
读写	uint32_t	GainB	校准参数 B 值
读写	uint32_t	CalTimes	校准倍数
读写	uint32_t	CalFactor	校准因子

3.8.1 HAL_ADC_Init

3.8.1.1 功能介绍

初始化 ADC。

3.8.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Init (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.2 HAL_ADC_DeInit

3.8.2.1 功能介绍

清除 ADC 初始化参数。

3.8.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_DeInit (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.3 HAL_ADC_MspInit

3.8.3.1 功能介绍

初始化 ADC Msp。

3.8.3.2 接口定义

函数接口	void HAL_ADC_MspInit (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.4 HAL_ADC_MspDeInit

3.8.4.1 功能介绍

清除 ADC Msp 初始化参数。

3.8.4.2 接口定义

函数接口	void HAL_ADC_MspDeInit (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.5 HAL_ADC_Start

3.8.5.1 功能介绍

使能 ADC，启动转换。

3.8.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Start (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.6 HAL_ADC_Stop

3.8.6.1 功能介绍

禁止使能 ADC，停止转换。

3.8.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Stop (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.7 HAL_ADC_PollForConversion

3.8.7.1 功能介绍

等待规则组转换完成。

3.8.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_PollForConversion (ADC_HandleTypeDef * hadc, uint32_t Timeout)
输入	ADC_HandleTypeDef * hadc uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.8 HAL_ADC_PollForEvent

3.8.8.1 功能介绍

轮询 ADC 事件。

3.8.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_PollForEvent (ADC_HandleTypeDef * hadc, uint32_t EventType, uint32_t Timeout)
输入	ADC_HandleTypeDef * hadc uint32_t EventType: { ADC_AWD1_EVENT, ADC_OVR_EVENT } uint32_t Timeout

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.9 HAL_ADC_Start_IT

3.8.9.1 功能介绍

开启 ADC，开启带中断的规则组转换。

3.8.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Start_IT (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.10 HAL_ADC_Stop_IT

3.8.10.1 功能介绍

关闭 ADC，停止带中断的规则组转换。

3.8.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Stop_IT (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.11 HAL_ADC_Start_DMA

3.8.11.1 功能介绍

开启 ADC，开启带 DMA 的规则组转换。

3.8.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Start_DMA (ADC_HandleTypeDef * hadc, uint32_t * pData, uint32_t Length)
输入	ADC_HandleTypeDef * hadc uint32_t * pData uint32_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.12 HAL_ADC_Stop_DMA

3.8.12.1 功能介绍

关闭 ADC，停止带 DMA 的规则组转换。

3.8.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_Stop_DMA (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.13 HAL_ADC_GetValue

3.8.13.1 功能介绍

获取 ADC 规则组转换结果。

3.8.13.2 接口定义

函数接口	uint32_t HAL_ADC_GetValue (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	规则组转换数据
资源使用	
说明	

3.8.14 HAL_ADC_IRQHandler

3.8.14.1 功能介绍

处理 ADC 中断请求。

3.8.14.2 接口定义

函数接口	void HAL_ADC_IRQHandler (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.15 HAL_ADC_ConvCpltCallback

3.8.15.1 功能介绍

在非阻塞模式下转换完成回调。

3.8.15.2 接口定义

函数接口	void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc

输出	无
返回值	无
资源使用	
说明	

3.8.16 HAL_ADC_ConvHalfCpltCallback

3.8.16.1 功能介绍

在非阻塞模式下转换一半回调。

3.8.16.2 接口定义

函数接口	void HAL_ADC_ConvHalfCpltCallback (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.17 HAL_ADC_LevelOutOfWindowCallback

3.8.17.1 功能介绍

在非阻塞模式下模拟看门狗回调。

3.8.17.2 接口定义

函数接口	void HAL_ADC_LevelOutOfWindowCallback (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.18 HAL_ADC_ErrorCallback

3.8.18.1 功能介绍

在非阻塞模式下错误回调。

3.8.18.2 接口定义

函数接口	void HAL_ADC_ErrorCallback (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.19 HAL_ADC_ConfigChannel

3.8.19.1 功能介绍

配置 ADC 通道。

3.8.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_ConfigChannel (ADC_HandleTypeDef * hadc, ADC_ChannelConfTypeDef * pConfig)
输入	ADC_HandleTypeDef * hadc ADC_ChannelConfTypeDef * pConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.20 HAL_ADC_AnalogWDGConfig

3.8.20.1 功能介绍

配置 ADC 模拟看门狗。

3.8.20.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADC_AnalogWDGConfig (ADC_HandleTypeDef * hadc, ADC_AnalogWDGConfTypeDef * pAnalogWDGConfig)
输入	ADC_HandleTypeDef * hadc ADC_AnalogWDGConfTypeDef * pAnalogWDGConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.21 HAL_ADC_GetState

3.8.21.1 功能介绍

配置 ADC 状态。

3.8.21.2 接口定义

函数接口	uint32_t HAL_ADC_GetState (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	ADC handle state
资源使用	
说明	

3.8.22 HAL_ADC_GetError

3.8.22.1 功能介绍

配置 ADC 错误代码。

3.8.22.2 接口定义

函数接口	uint32_t HAL_ADC_GetError (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无

返回值	ADC error code
资源使用	
说明	

3.8.23 HAL_ADCEx_Calibration_Start

3.8.23.1 功能介绍

开启 ADC 自校准。

3.8.23.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADCEx_Calibration_Start (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.24 HAL_ADCEx_Calibration_GetValue

3.8.24.1 功能介绍

获取校准因子。

3.8.24.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADCEx_Calibration_GetValue (ADC_HandleTypeDef * hadc, ADC_CalibrationConfTypeDef * pCalConfig)
输入	ADC_HandleTypeDef * hadc ADC_CalibrationConfTypeDef * pCalConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.8.25 HAL_ADCEx_Calibration_SetValue

3.8.25.1 功能介绍

设置校准因子。

3.8.25.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ADCEx_Calibration_SetValue (ADC_HandleTypeDef * hadc, ADC_CalibrationConfTypeDef * pCalConfig)
输入	ADC_HandleTypeDef * hadc ADC_CalibrationConfTypeDef * pCalConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.8.26 HAL_ADCEx_ChannelConfigReadyCallback

3.8.26.1 功能介绍

在非阻塞模式下 ADC 通道配置准备回调。

3.8.26.2 接口定义

函数接口	void HAL_ADCEx_ChannelConfigReadyCallback (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.8.27 HAL_ADCEx_ChannelCalibrationReadyCallback

3.8.27.1 功能介绍

在非阻塞模式下 ADC 校准准备回调。

3.8.27.2 接口定义

函数接口	void HAL_ADCEx_ChannelCalibrationReadyCallback (ADC_HandleTypeDef * hadc)
输入	ADC_HandleTypeDef * hadc
输出	无
返回值	无
资源使用	
说明	

3.9 ATK 模块

宏定义	数值	含义
HAL_ATK_StateTypeDef		
HAL_ATK_STATE_RESET	0x00	重置
HAL_ATK_STATE_READY	0x01	就绪
HAL_ATK_STATE_BUSY	0x02	忙
HAL_ATK_STATE_TIMEOUT	0x03	超时
HAL_ATK_STATE_ERROR	0x04	错误
HAL_ATK_STATE_ERROR_DMA	0x05	DMA 错误

属性	类型	字段名	含义
ATK_TypeDef			
读写	uint32_t	CRA	TKA 控制

			寄存器
读写	uint32_t	CRB	TKB 控制寄存器
读写	uint32_t	ISR	TK 状态寄存器
读写	uint32_t	CHS	TK 启动通道选择寄存器
读写	uint32_t	CHAEN	TKA 通道使能寄存器
读写	uint32_t	CHBEN	TKB 通道使能寄存器
读写	uint32_t	ITRIMA1	TKA 通道调试参数 1
读写	uint32_t	ITRIMA2	TKA 通道调试参数 2
读写	uint32_t	ITRIMA3	TKA 通道调试参数 3
读写	uint32_t	ITRIMA4	TKA 通道调试参数 4
只读	uint32_t	ITRIMA5	TKA 通道调试参数 5
读写	uint32_t	ITRIMA6	TKA 通道调试参数 6
读写	uint32_t	ITRIMB1	TKB 通道调试参数 1
读写	uint32_t	ITRIMB2	TKB 通道调试参数 2
读写	uint32_t	ITRIMB3	TKB 通道调试参数 3
读写	uint32_t	ITRIMB4	TKB 通道调试参数 4
读写	uint32_t	ITRIMB5	TKB 通道调试参数 5
读写	uint32_t	ITRIMB6	TKB 通道

			调试参数 6
只读	uint32_t	RAMA	TKA RAM 1~20
只读	uint32_t	RAMB	TKB RAM 1~20
ATK_InitTypeDef			
读写	uint32_t	ScanLength	扫描长度
读写	FunctionalState	ClockSpreadEnable	时钟展频使 能
读写	uint32_t	ClockTrimmimng	时钟校准值
读写	FunctionalState	ClockTrimmimngEnable	时钟校准使 能
读写	FunctionalState	XCapEnable	外部电容使 能
读写	FunctionalState	AutoRunEnable	自动运行使 能
读写	FunctionalState	AcshieldEnable	屏蔽使能
读写	uint32_t	VoltageReference	参考电压
读写	uint32_t	Mode	模式
读写	uint32_t	PortNum	端口号
ATK_HandleTypeDef			
读写	ATK_TypeDef	Instance	寄存器基础 地址
读写	ATK_InitTypeDef	Init	初始化参数
读写	DMA_HandleTypeDef	DMA_Handle	DMA 句柄
读写	HAL_StatusTypeDef	Status	状态
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_ATK_StateTypeDef	State	ATK 状态
读写	指针函数	void(*MspInitCallback)(struct __ATK_HandleTypeDef *hatk)	MSP 初始化 回调
读写	指针函数	void(*MspDeInitCallback)(struct __ATK_HandleTypeDef *hatk)	MSP 非初始 化回调
读写	指针函数	void (* ConvCpltCallback)(struct __ATK_HandleTypeDef *hatk)	传输完成回 调
读写	指针函数	void(*ConvHalfCpltCallback)(struct __ATK_HandleTypeDef *hatk)	传输完成一

			半回调
读写	指针函数	void(*ErrorCallback)(struct __ATK_HandleTypeDef *hatk)	错误回调

3.9.1 HAL_ATK_Init

3.9.1.1 功能介绍

ATK 初始化。

3.9.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Init (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.2 HAL_ATK_DeInit

3.9.2.1 功能介绍

清除 ATK 初始化参数。

3.9.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_DeInit (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.3 HAL_ATK_MspInit

3.9.3.1 功能介绍

ATK MSP 初始化。

3.9.3.2 接口定义

函数接口	void HAL_ATK_MspInit (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	无
资源使用	
说明	

3.9.4 HAL_ATK_MspDeInit

3.9.4.1 功能介绍

清除 ATK MSP 初始化参数。

3.9.4.2 接口定义

函数接口	void HAL_ATK_MspDeInit (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	无
资源使用	
说明	

3.9.5 HAL_ATK_Start

3.9.5.1 功能介绍

开启 ATK 的生成。

3.9.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Start (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.6 HAL_ATK_Stop

3.9.6.1 功能介绍

停止 ATK 的生成。

3.9.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Stop (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.7 HAL_ATK_Start_IT

3.9.7.1 功能介绍

以中断模式开启 ATK 的生成。

3.9.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Start_IT (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.8 HAL_ATK_Stop_IT

3.9.8.1 功能介绍

以中断模式停止 ATK 的生成。

3.9.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Stop_IT (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.9 HAL_ATK_Start_DMA

3.9.9.1 功能介绍

以 DMA 模式开启 ATK 的生成。

3.9.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Start_DMA (ATK_HandleTypeDef * hatk, uint16_t * pData, uint32_t Length)
输入	ATK_HandleTypeDef * hatk uint16_t * pData uint32_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.10 HAL_ATK_Stop_DMA

3.9.10.1 功能介绍

以 DMA 模式停止 ATK 的生成。

3.9.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_Stop_DMA (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.11 HAL_ATK_GetValue

3.9.11.1 功能介绍

获取当前端口通道值。

3.9.11.2 接口定义

函数接口	uint32_t HAL_ATK_GetValue (ATK_HandleTypeDef * hatk, uint32_t index)
输入	ATK_HandleTypeDef * hatk uint32_t index: [0, 39]
输出	无
返回值	通道值
资源使用	
说明	

3.9.12 HAL_ATK_ConfigChannel

3.9.12.1 功能介绍

配置分配给 ATK 端口的通道。

3.9.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_ConfigChannel (ATK_HandleTypeDef * hatk, ATK_ChannelConfTypeDef * pConfig)
输入	ATK_HandleTypeDef * hatk ATK_ChannelConfTypeDef * pConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.13 HAL_ATK_ConfigFirstChannel

3.9.13.1 功能介绍

配置要分配给 ATK 端口的第一个通道。

3.9.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_ATK_ConfigFirstChannel (ATK_HandleTypeDef * hatk, uint32_t channel)
输入	ATK_HandleTypeDef * hatk uint32_t channel
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.9.14 HAL_ATK_IRQHandler

3.9.14.1 功能介绍

处理 ATK 中断请求。

3.9.14.2 接口定义

函数接口	void HAL_ATK_IRQHandler (ATK_HandleTypeDef * hatk)
------	--

输入	ATK_HandleTypeDef * hatk
输出	无
返回值	无
资源使用	
说明	

3.9.15 HAL_ATK_ConvCpltCallback

3.9.15.1 功能介绍

转换完成回调。

3.9.15.2 接口定义

函数接口	void HAL_ATK_ConvCpltCallback (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	无
资源使用	
说明	

3.9.16 HAL_ATK_ConvHalfCpltCallback

3.9.16.1 功能介绍

转换完成一半回调。

3.9.16.2 接口定义

函数接口	void HAL_ATK_ConvHalfCpltCallback (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	无
资源使用	
说明	

3.9.17 HAL_ATK_ErrorCallback

3.9.17.1 功能介绍

错误回调。

3.9.17.2 接口定义

函数接口	void HAL_ATK_ErrorCallback (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	无
资源使用	
说明	

3.9.18 HAL_ATK_GetState

3.9.18.1 功能介绍

获取 ATK 状态。

3.9.18.2 接口定义

函数接口	HAL_ATK_StateTypeDef HAL_ATK_GetState (ATK_HandleTypeDef * hatk)
输入	ATK_HandleTypeDef * hatk
输出	无
返回值	HAL_ATK_StateTypeDef
资源使用	
说明	

3.10 COMP 模块

宏定义	数值	含义
HAL_COMP_StateTypeDef		
HAL_ATK_STATE_RESET	0x00	重置
HAL_COMP_STATE_RESET_LOCKED	0x00 0x10	重置且配置锁定
HAL_ATK_STATE_READY	0x01	就绪
HAL_COMP_STATE_READY_LOCKED	0x01 0x10	就绪且配置锁定
HAL_ATK_STATE_BUSY	0x02	忙
HAL_COMP_STATE_BUSY_LOCKED	0x00 0x10	忙且配置锁定

属性	类型	字段名	含义
COMP_TypeDef			
读写	uint32_t	CSR	控制 和状态寄存器
COMP_InitTypeDef			
读写	uint32_t	WindowMode	窗口比较器模式
读写	uint32_t	WindowOutput	窗口模式输出
读写	uint32_t	InputPlus	正相输入信号
读写	uint32_t	InputMinus	反相输入信号
读写	uint32_t	OutputPol	输出极性

读写	uint32_t	TriggerMode	触发模式
读写	FunctionalState	OutputFilterEnable	输出滤波使能
读写	uint32_t	OutputFilterTimes	输出滤波时间
COMP_HandleTypeDef			
读写	COMP_TypeDef	Instance	寄存器基础地址
读写	COMP_InitTypeDef	Init	初始化参数
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_COMP_StateTypeDef	State	COMP 状态
读写	uint32_t	ErrorCode	错误代码
读写	指针函数	void(*TriggerCallback)(struct __COMP_HandleTypeDef *hcomp)	触发回调
读写	指针函数	void(*MspInitCallback)(struct __COMP_HandleTypeDef *hcomp)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback)(struct __COMP_HandleTypeDef *hcomp)	MSP 非初始化回调

3.10.1 HAL_COMP_Init

3.10.1.1 功能介绍

初始化 COMP。

3.10.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_COMP_Init (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.10.2 HAL_COMP_DeInit

3.10.2.1 功能介绍

清除 COMP 初始化参数。

3.10.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_COMP_DeInit (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.10.3 HAL_COMP_MspInit

3.10.3.1 功能介绍

初始化 COMP MSP。

3.10.3.2 接口定义

函数接口	void HAL_COMP_MspInit (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	无
资源使用	
说明	

3.10.4 HAL_COMP_MspDeInit

3.10.4.1 功能介绍

清除 COMP 初始化参数。

3.10.4.2 接口定义

函数接口	void HAL_COMP_MspDeInit (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	无
资源使用	
说明	

3.10.5 HAL_COMP_Start

3.10.5.1 功能介绍

开启 COMP。

3.10.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_COMP_Start (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.10.6 HAL_COMP_Stop

3.10.6.1 功能介绍

关闭 COMP。

3.10.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_COMP_Stop (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.10.7 HAL_COMP_IRQHandler

3.10.7.1 功能介绍

处理 COMP 中断请求。

3.10.7.2 接口定义

函数接口	void HAL_COMP_IRQHandler (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	无
资源使用	
说明	

3.10.8 HAL_COMP_Lock

3.10.8.1 功能介绍

锁定选定的比较器配置。

3.10.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_COMP_Lock (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.10.9 HAL_COMP_GetOutputLevel

3.10.9.1 功能介绍

获取选定的比较器输出水平。

3.10.9.2 接口定义

函数接口	uint32_t HAL_COMP_GetOutputLevel (COMP_HandleTypeDef * hcomp)
------	---

	hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	{ COMP_OUTPUT_LEVEL_LOW, COMP_OUTPUT_LEVEL_HIGH }
资源使用	
说明	

3.10.10 HAL_COMP_TriggerCallback

3.10.10.1 功能介绍

比较器触发回调。

3.10.10.2 接口定义

函数接口	void HAL_COMP_TriggerCallback (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	无
资源使用	
说明	

3.10.11 HAL_COMP_GetState

3.10.11.1 功能介绍

获取比较器状态。

3.10.11.2 接口定义

函数接口	HAL_COMP_StateTypeDef HAL_COMP_GetState (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	HAL_COMP_StateTypeDef
资源使用	
说明	

3.10.12 HAL_COMP_GetError

3.10.12.1 功能介绍

获取比较器状态。

3.10.12.2 接口定义

函数接口	uint32_t HAL_COMP_GetError (COMP_HandleTypeDef * hcomp)
输入	COMP_HandleTypeDef * hcomp
输出	无
返回值	错误值
资源使用	
说明	

3.11 CRC 模块

宏定义	数值	含义
HAL_CRC_StateTypeDef		
HAL_CRC_STATE_RESET	0x00	重置
HAL_CRC_STATE_READY	0x01	就绪
HAL_CRC_STATE_BUSY	0x02	忙
HAL_CRC_STATE_TIMEOUT	0x03	超时
HAL_CRC_STATE_ERROR	0x04	错误

属性	类型	字段名	含义
CRC_TypeDef			
读写	uint32_t	CR	控制状态寄存器
读写	uint32_t	RR	结果寄存器
读写	uint32_t	DR	数据寄存器
CRC_InitTypeDef			
读写	uint32_t	CRCLength	CRC 长度
CRC_HandleTypeDef			
读写	CRC_TypeDef	Instance	寄存器基础地址
读写	CRC_InitTypeDef	Init	初始化参数
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_CRC_StateTypeDef	State	CRC 状态
读写	uint32_t	InputDataFormat	输出数据格式

3.11.1 HAL_CRC_Init

3.11.1.1 功能介绍

初始化 CRC。

3.11.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CRC_Init (CRC_HandleTypeDef * hrcr)
输入	CRC_HandleTypeDef * hrcr
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.11.2 HAL_CRC_DeInit

3.11.2.1 功能介绍

清除 CRC 初始化参数。

3.11.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CRC_DeInit (CRC_HandleTypeDef * hrcr)
输入	CRC_HandleTypeDef * hrcr
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.11.3 HAL_CRC_MspInit

3.11.3.1 功能介绍

初始化 CRC MSP。

3.11.3.2 接口定义

函数接口	void HAL_CRC_MspInit (CRC_HandleTypeDef * hrcr)
输入	CRC_HandleTypeDef * hrcr
输出	无
返回值	无
资源使用	
说明	

3.11.4 HAL_CRC_MspDeInit

3.11.4.1 功能介绍

清除 CRC MSP 初始化参数。

3.11.4.2 接口定义

函数接口	void HAL_CRC_MspDeInit (CRC_HandleTypeDef * hrcr)
输入	CRC_HandleTypeDef * hrcr
输出	无
返回值	无
资源使用	
说明	

3.11.5 HAL_CRC_Accumulate

3.11.5.1 功能介绍

以先前计算的 CRC 作为初始值开始，计算 8 位数据缓冲区的 8、16 或 32 位 CRC 值。

3.11.5.2 接口定义

函数接口	uint32_t HAL_CRC_Accumulate (CRC_HandleTypeDef * hrcr, uint32_t pBuffer[], uint32_t BufferLength)
输入	CRC_HandleTypeDef * hrcr uint32_t pBuffer[] uint32_t BufferLength
输出	无
返回值	CRC 计算值

资源使用	
说明	

3.11.6 HAL_CRC_Calculate

3.11.6.1 功能介绍

从初始化值开始计算 8 位数据缓冲区的 8、16 或 32 位 CRC 值。

3.11.6.2 接口定义

函数接口	uint32_t HAL_CRC_Calculate (CRC_HandleTypeDef * hrcr, uint32_t pBuffer[], uint32_t BufferLength)
输入	CRC_HandleTypeDef * hrcr uint32_t pBuffer[] uint32_t BufferLength
输出	无
返回值	CRC 计算值
资源使用	
说明	

3.11.7 HAL_CRC_CompareCalculate

3.11.7.1 功能介绍

计算作为初始化值开始的 8、16 或 32 位数据缓冲区的 8、16 或 32 位 CRC 值，再与正确结果进行比较。

3.11.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CRC_CompareCalculate (CRC_HandleTypeDef * hrcr, uint32_t pBuffer[], uint32_t BufferLength, uint32_t res)
输入	CRC_HandleTypeDef * hrcr uint32_t pBuffer[] uint32_t BufferLength uint32_t res
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.11.8 HAL_CRC_GetState

3.11.8.1 功能介绍

获取 CRC 状态。

3.11.8.2 接口定义

函数接口	HAL_CRC_StateTypeDef HAL_CRC_GetState (CRC_HandleTypeDef * hrcr)
输入	CRC_HandleTypeDef * hrcr
输出	无

返回值	HAL_CRC_StateTypeDef
资源使用	
说明	

3.12 DAC 模块

宏定义	数值	含义
HAL_DAC_StateTypeDef		
HAL_DAC_STATE_RESET	0x00	重置
HAL_DAC_STATE_READY	0x01	就绪
HAL_DAC_STATE_BUSY	0x02	忙
HAL_DAC_STATE_TIMEOUT	0x03	超时
HAL_DAC_STATE_ERROR	0x04	错误

属性	类型	字段名	含义
DAC_TypeDef			
读写	uint32_t	CR	控制寄存器
只写	uint32_t	SWTRIGR	软件触发寄存器
读写	uint32_t	DHR12R1	DAC1 通道 12 位右对齐数据保持寄存器
读写	uint32_t	DHR12L1	DAC1 通道 12 位左对齐数据保持寄存器
读写	uint32_t	DHR8R1	DAC1 通道 8 位右对齐数据保持寄存器
读写	uint32_t	DHR12R2	DAC2 通

			道 12 位右 对齐数据 保持寄存 器
读写	uint32_t	DHR12L2	DAC2 通 道 12 位左 对齐数据 保持寄存 器
读写	uint32_t	DHR8R2	DAC2 通 道 8 位右 对齐数据 保持寄存 器
读写	uint32_t	DHR12RD	DAC 双通 道 12 位右 对齐数据 保持寄存 器
读写	uint32_t	DHR12LD	DAC 双通 道 12 位左 对齐数据 保持寄存 器
读写	uint32_t	DHR8RD	DAC 双通 道 8 位右 对齐数据 保持寄存 器
只读	uint32_t	DOR1	DAC1 通 道数据输 出寄存器
只读	uint32_t	DOR2	DAC2 通 道数据输 出寄存器

读写	uint32_t	SR	状态寄存器
读写	uint32_t	MCR	模式控制寄存器
读写	uint32_t	CR2	控制寄存器 2
只写	uint32_t	SWTR2	软件触发寄存器 2
读写	uint32_t	DHR12R3	DAC3 通道 12 位右对齐数据保持寄存器
读写	uint32_t	DHR12R4	DAC4 通道 12 位右对齐数据保持寄存器
只读	uint32_t	DOR3	DAC3 通道数据输出寄存器
只读	uint32_t	DOR4	DAC4 通道数据输出寄存器
读写	uint32_t	SR2	状态寄存器 2
读写	uint32_t	MCR2	模式控制寄存器 2
DAC_HandleTypeDef			
读写	DAC_TypeDef	Instance	寄存器地址
读写	HAL_DAC_StateTypeDef	Init	DAC 状态
读写	HAL_LockTypeDef	Lock	锁定
读写	DMA_HandleTypeDef	State	DMA 句柄
读写	uint32_t	ErrorCode	错误代码

读写	指针函数	void(*ConvCpltCallbackCh1)(struct __DAC_HandleTypeDef *hdac)	通道 1 传输完成回调
读写	指针函数	void(*ConvHalfCpltCallbackCh1)(struct __DAC_HandleTypeDef *hdac)	通道 1 传输完成一半回调
读写	指针函数	void(*ErrorCallbackCh1)(struct __DAC_HandleTypeDef *hdac)	通道 1 错误回调
读写	指针函数	void(*DMAUnderrunCallbackCh1)(struct __DAC_HandleTypeDef *hdac)	通道 1 DMA 下溢回调
读写	指针函数	void(*ConvCpltCallbackCh2)(struct __DAC_HandleTypeDef *hdac)	通道 2 传输完成回调
读写	指针函数	void(*ConvHalfCpltCallbackCh2)(struct __DAC_HandleTypeDef *hdac)	通道 2 传输完成一半回调
读写	指针函数	void(*ErrorCallbackCh2)(struct __DAC_HandleTypeDef *hdac)	通道 2 错误回调
读写	指针函数	void(*DMAUnderrunCallbackCh2)(struct __DAC_HandleTypeDef *hdac)	通道 2 DMA 下溢回调
读写	指针函数	void(*ConvCpltCallbackCh3)(struct __DAC_HandleTypeDef *hdac)	通道 3 传输完成回调
读写	指针函数	void(*ConvHalfCpltCallbackCh3)(struct __DAC_HandleTypeDef *hdac)	通道 3 传输完成一半回调
读写	指针函数	void(*ErrorCallbackCh3)(struct __DAC_HandleTypeDef *hdac)	通道 3 错误回调
读写	指针函数	void(*DMAUnderrunCallbackCh3)(struct __DAC_HandleTypeDef *hdac)	通道 3 DMA 下溢回调
读写	指针函数	void(*ConvCpltCallbackCh4)(struct __DAC_HandleTypeDef *hdac)	通道 4 传输完成回

			调
读写	指针函数	void(*ConvHalfCpltCallbackCh4)(struct __DAC_HandleTypeDef *hdac)	通道 4 传输完成一半回调
读写	指针函数	void(*ErrorCallbackCh4)(struct __DAC_HandleTypeDef *hdac)	通道 4 错误回调
读写	指针函数	void(*DMAUnderrunCallbackCh4)(struct __DAC_HandleTypeDef *hdac)	通道 4 DMA 下溢回调
读写	指针函数	void(*MspInitCallback)(struct __DAC_HandleTypeDef *hdac)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback)(struct __DAC_HandleTypeDef *hdac)	MSP 非初始化回调

3.12.1 HAL_DAC_Init

3.12.1.1 功能介绍

DAC 初始化。

3.12.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_Init (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.2 HAL_DAC_DeInit

3.12.2.1 功能介绍

清除 DAC 初始化参数。

3.12.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_DeInit (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.3 HAL_DAC_MspInit

3.12.3.1 功能介绍

DAC MSP 初始化。

3.12.3.2 接口定义

函数接口	void HAL_DAC_MspInit (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.4 HAL_DAC_MspDeInit

3.12.4.1 功能介绍

清除 DAC MSP 初始化参数。

3.12.4.2 接口定义

函数接口	void HAL_DAC_MspDeInit (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.5 HAL_DAC_Start

3.12.5.1 功能介绍

开启 DAC 通道转换。

3.12.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_Start (DAC_HandleTypeDef * hdac, uint32_t Channel)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

3.12.6 HAL_DAC_Stop

3.12.6.1 功能介绍

停止 DAC 通道转换。

3.12.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_Stop (DAC_HandleTypeDef * hdac, uint32_t Channel)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel:

	{ LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	无
资源使用	
说明	

3.12.7 HAL_DAC_Start_DMA

3.12.7.1 功能介绍

在 DMA 模式下，开启 DAC 通道转换。

3.12.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_Start_DMA (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t * pData, uint32_t Length, uint32_t Alignment)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t * pData uint32_t Length uint32_t Alignment: { DAC_ALIGN_8B_R, DAC_ALIGN_12B_L, DAC_ALIGN_12B_R }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.8 HAL_DAC_Stop_DMA

3.12.8.1 功能介绍

在 DMA 模式下，停止 DAC 通道转换。

3.12.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_Stop_DMA (DAC_HandleTypeDef * hdac, uint32_t Channel)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.9 HAL_DAC_IRQHandler

3.12.9.1 功能介绍

处理 DAC 中断请求。

3.12.9.2 接口定义

函数接口	void HAL_DAC_IRQHandler (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.10 HAL_DAC_SetValue

3.12.10.1 功能介绍

设置 DAC 通道的指定数据保存寄存器值。

3.12.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_SetValue (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Alignment, uint32_t Data)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Alignment: { DAC_ALIGN_8B_R, DAC_ALIGN_12B_L, DAC_ALIGN_12B_R } uint32_t Data
输出	无
返回值	无
资源使用	
说明	

3.12.11 HAL_DAC_ConvCpltCallbackCh1

3.12.11.1 功能介绍

设置 DAC 通道 1 在非阻塞模式转换完成回调。

3.12.11.2 接口定义

函数接口	void HAL_DAC_ConvCpltCallbackCh1 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.12 HAL_DAC_ConvHalfCpltCallbackCh1

3.12.12.1 功能介绍

设置 DAC 通道 1 在非阻塞模式转换完成一半回调。

3.12.12.2 接口定义

函数接口	void HAL_DAC_ConvHalfCpltCallbackCh1 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.13 HAL_DAC_ErrorCallbackCh1

3.12.13.1 功能介绍

设置 DAC 通道 1 在错误回调。

3.12.13.2 接口定义

函数接口	void HAL_DAC_ErrorCallbackCh1 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.14 HAL_DAC_DMAUnderrunCallbackCh1

3.12.14.1 功能介绍

设置 DAC 通道 1 DMA 溢出回调。

3.12.14.2 接口定义

函数接口	void HAL_DAC_DMAUnderrunCallbackCh1 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.15 HAL_DAC_GetValue

3.12.15.1 功能介绍

获取 DAC 通道的最后一个数据输出值。

3.12.15.2 接口定义

函数接口	uint32_t HAL_DAC_GetValue (DAC_HandleTypeDef * hdac, uint32_t Channel)
------	--

输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	数据输出值
资源使用	
说明	

3.12.16 HAL_DAC_ConfigChannel

3.12.16.1 功能介绍

配置 DAC 通道。

3.12.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DAC_ConfigChannel (DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel)
输入	DAC_HandleTypeDef * hdac DAC_ChannelConfTypeDef * sConfig uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.17 HAL_DAC_GetState

3.12.17.1 功能介绍

获取 DAC 状态。

3.12.17.2 接口定义

函数接口	HAL_DAC_StateTypeDef HAL_DAC_GetState (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	HAL_DAC_StateTypeDef
资源使用	
说明	

3.12.18 HAL_DAC_GetError

3.12.18.1 功能介绍

获取 DAC 错误。

3.12.18.2 接口定义

函数接口	uint32_t HAL_DAC_GetError (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	错误值
资源使用	
说明	

3.12.19 HAL_DACEx_TriangleWaveGenerate

3.12.19.1 功能介绍

启用或禁用 DAC 通道三角波生成功能。

3.12.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DACEx_TriangleWaveGenerate (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Amplitude)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Amplitude: { DAC_TRIANGLEAMPLITUDE_1, DAC_TRIANGLEAMPLITUDE_3, DAC_TRIANGLEAMPLITUDE_7, DAC_TRIANGLEAMPLITUDE_15, DAC_TRIANGLEAMPLITUDE_31, DAC_TRIANGLEAMPLITUDE_63, DAC_TRIANGLEAMPLITUDE_127, DAC_TRIANGLEAMPLITUDE_255, DAC_TRIANGLEAMPLITUDE_511, DAC_TRIANGLEAMPLITUDE_1023, DAC_TRIANGLEAMPLITUDE_2047, DAC_TRIANGLEAMPLITUDE_4095 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.20 HAL_DACEx_NoiseWaveGenerate

3.12.20.1 功能介绍

启用或禁用 DAC 通道噪声波生成功能。

3.12.20.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DACEx_NoiseWaveGenerate (DAC_HandleTypeDef * hdac, uint32_t Channel, uint32_t Amplitude)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2,

	LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Amplitude: { DAC_LFSRUNMASK_BIT0, DAC_LFSRUNMASK_BITS1_0, DAC_LFSRUNMASK_BITS2_0, DAC_LFSRUNMASK_BITS3_0, DAC_LFSRUNMASK_BITS4_0, DAC_LFSRUNMASK_BITS5_0, DAC_LFSRUNMASK_BITS6_0, DAC_LFSRUNMASK_BITS7_0, DAC_LFSRUNMASK_BITS8_0, DAC_LFSRUNMASK_BITS9_0, DAC_LFSRUNMASK_BITS10_0, DAC_LFSRUNMASK_BITS11_0}
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.21 HAL_DACEx_DualSetValue

3.12.21.1 功能介绍

为双 DAC 通道设置指定的数据保存寄存器值。

3.12.21.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DACEx_DualSetValue (DAC_HandleTypeDef * hdac, uint32_t Alignment, uint32_t Data1, uint32_t Data2)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t Alignment: { DAC_ALIGN_8B_R, DAC_ALIGN_12B_L, DAC_ALIGN_12B_R } uint32_t Data1 uint32_t Data2
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.22 HAL_DACEx_DualGetValue

3.12.22.1 功能介绍

获取 DAC 通道的最后一个数据输出值。

3.12.22.2 接口定义

函数接口	uint32_t HAL_DACEx_DualGetValue (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	HAL_StatusTypeDef
资源使用	数据输出值
说明	

3.12.23 HAL_DACEx_ConvCpltCallbackCh2

3.12.23.1 功能介绍

设置 DAC 通道 2 在非阻塞模式转换完成回调。

3.12.23.2 接口定义

函数接口	void HAL_DACEx_ConvCpltCallbackCh2 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.24 HAL_DACEx_ConvHalfCpltCallbackCh2

3.12.24.1 功能介绍

设置 DAC 通道 2 在非阻塞模式转换完成一半回调。

3.12.24.2 接口定义

函数接口	void HAL_DACEx_ConvHalfCpltCallbackCh2 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.25 HAL_DACEx_ErrorCallbackCh2

3.12.25.1 功能介绍

设置 DAC 通道 2 在错误回调。

3.12.25.2 接口定义

函数接口	void HAL_DACEx_ErrorCallbackCh2 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.26 HAL_DACEx_DMAUnderrunCallbackCh2

3.12.26.1 功能介绍

设置 DAC 通道 2 DMA 溢出回调。

3.12.26.2 接口定义

函数接口	void HAL_DACEx_DMAUnderrunCallbackCh2 (DAC_HandleTypeDef * hdac)
------	--

输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.27 HAL_DACEx_ConvCpltCallbackCh3

3.12.27.1 功能介绍

设置 DAC 通道 3 在非阻塞模式转换完成回调。

3.12.27.2 接口定义

函数接口	void HAL_DACEx_ConvCpltCallbackCh3 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.28 HAL_DACEx_ConvHalfCpltCallbackCh3

3.12.28.1 功能介绍

设置 DAC 通道 3 在非阻塞模式转换完成一半回调。

3.12.28.2 接口定义

函数接口	void HAL_DACEx_ConvHalfCpltCallbackCh3 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.29 HAL_DACEx_ErrorCallbackCh3

3.12.29.1 功能介绍

设置 DAC 通道 3 在错误回调。

3.12.29.2 接口定义

函数接口	void HAL_DACEx_ErrorCallbackCh3 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.30 HAL_DACEx_DMAUnderrunCallbackCh3

3.12.30.1 功能介绍

设置 DAC 通道 3 DMA 溢出回调。

3.12.30.2 接口定义

函数接口	void HAL_DACEx_DMAUnderrunCallbackCh3 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.31 HAL_DACEx_ConvCpltCallbackCh4

3.12.31.1 功能介绍

设置 DAC 通道 4 在非阻塞模式转换完成回调。

3.12.31.2 接口定义

函数接口	void HAL_DACEx_ConvCpltCallbackCh4 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.32 HAL_DACEx_ConvHalfCpltCallbackCh4

3.12.32.1 功能介绍

设置 DAC 通道 4 在非阻塞模式转换完成一半回调。

3.12.32.2 接口定义

函数接口	void HAL_DACEx_ConvHalfCpltCallbackCh4 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.33 HAL_DACEx_ErrorCallbackCh4

3.12.33.1 功能介绍

设置 DAC 通道 4 在错误回调。

3.12.33.2 接口定义

函数接口	void HAL_DACEx_ErrorCallbackCh4(DAC_HandleTypeDef * hdac)
------	---

输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.34 HAL_DACEx_DMAUnderrunCallbackCh4

3.12.34.1 功能介绍

设置 DAC 通道 4DMA 溢出回调。

3.12.34.2 接口定义

函数接口	void HAL_DACEx_DMAUnderrunCallbackCh4 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.35 HAL_DACEx_DMAUnderrunCallbackCh4

3.12.35.1 功能介绍

设置 DAC 通道 4DMA 溢出回调。

3.12.35.2 接口定义

函数接口	void HAL_DACEx_DMAUnderrunCallbackCh4 (DAC_HandleTypeDef * hdac)
输入	DAC_HandleTypeDef * hdac
输出	无
返回值	无
资源使用	
说明	

3.12.36 HAL_DACEx_SelfCalibrate

3.12.36.1 功能介绍

设置 DAC 通道自校准。

3.12.36.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DACEx_SelfCalibrate (DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel)
输入	DAC_HandleTypeDef * hdac DAC_ChannelConfTypeDef * sConfig uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 }
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.37 HAL_DACEx_SetUserTrimming

3.12.37.1 功能介绍

设置 DAC 通道用户校准。

3.12.37.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DACEx_SetUserTrimming (DAC_HandleTypeDef * hdac, DAC_ChannelConfTypeDef * sConfig, uint32_t Channel, uint32_t NewTrimmingValue)
输入	DAC_HandleTypeDef * hdac DAC_ChannelConfTypeDef * sConfig uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t NewTrimmingValue
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.12.38 HAL_DACEx_GetTrimOffset

3.12.38.1 功能介绍

获取 DAC 通道校准值。

3.12.38.2 接口定义

函数接口	uint32_t HAL_DACEx_GetTrimOffset (DAC_HandleTypeDef * hdac, uint32_t Channel)
输入	DAC_HandleTypeDef * hdac uint32_t DAC_Channel: { LL_DAC_CHANNEL_1, LL_DAC_CHANNEL_2, LL_DAC_CHANNEL_3, LL_DAC_CHANNEL_4 } uint32_t NewTrimmingValue
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13 DMA 模块

宏定义	数值	含义
HAL_DMA_StateTypeDef		
HAL_DMA_STATE_RESET	0x00	重置

HAL_DMA_STATE_READY	0x01	就绪
HAL_DMA_STATE_BUSY	0x02	忙
HAL_DMA_STATE_TIMEOUT	0x03	超时

属性	类型	字段名	含义
DMA_TypeDef			
只读	uint32_t	ISR	中断状态寄存器
只写	uint32_t	ICR	中断标志清除寄存器
读写	uint32_t	CCR0	DMA 通道 0 控制寄存器
读写	uint32_t	CNDTR0	DMA 通道 0 待传输次数寄存器
读写	uint32_t	CSAR0	DMA 通道 0 源地址寄存器
读写	uint32_t	CDAR0	DMA 通道 0 目的地址寄存器
读写	uint32_t	CCR1	DMA 通道 1 控制寄存器
读写	uint32_t	CNDTR1	DMA 通道 1 待传输次数寄存器
读写	uint32_t	CSAR1	DMA 通道 1 源地址寄存器
读写	uint32_t	CDAR1	DMA 通道 1 目的地址寄存器
读写	uint32_t	CCR2	DMA 通道 2 控制寄存器
只读	uint32_t	CNDTR2	DMA 通道 2

			待传输次数寄存器
只读	uint32_t	CSAR2	DMA 通道 2 源地址寄存器
读写	uint32_t	CDAR2	DMA 通道 2 目的地址寄存器
读写	uint32_t	CCR3	DMA 通道 3 控制寄存器
读写	uint32_t	CNDTR3	DMA 通道 3 待传输次数寄存器
只写	uint32_t	CSAR3	DMA 通道 3 源地址寄存器
读写	uint32_t	CDAR3	DMA 通道 3 目的地址寄存器
DMA_InitTypeDef			
读写	uint32_t	ChannelIndex	源地址
读写	uint32_t	Request	外设请求
读写	uint32_t	SoftwareRequire	软件触发请求
读写	uint32_t	Mode	操作模式
读写	uint32_t	SrcInc	源地址递增模式
读写	uint32_t	DstInc	目的地址递增模式
读写	uint32_t	DataAlignment	传输数据对齐
读写	uint32_t	Priority	优先级
DMAMUX_TypeDef			
读写	uint32_t	C0CR	请求复用器通道 0 控制

			寄存器
读写	uint32_t	C1CR	请求复用器通道 1 控制寄存器
读写	uint32_t	C2CR	请求复用器通道 2 控制寄存器
读写	uint32_t	C3CR	请求复用器通道 3 控制寄存器
读写	uint32_t	RG0CR	请求生成器 0 控制寄存器
读写	uint32_t	RG1CR	请求生成器 1 控制寄存器
只读	uint32_t	RGSR	请求生成器中断状态寄存器
只写	uint32_t	RGCFR	请求生成器中断标志清除寄存器
DMA_HandleTypeDef			
读写	DMA_TypeDef	Instance	寄存器地址
读写	DMA_InitTypeDef	Init	初始化参数
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_DMA_StateTypeDef	State	状态
读写	指针函数	void *Parent	父对象状态
读写	指针函数	void(*XferCpltCallback)(struct __DMA_HandleTypeDef *hdma)	传输完成回调
读写	指针函数	void(*XferHalfCpltCallback)(struct __DMA_HandleTypeDef *hdma)	传输完成一半会滴
读写	指针函数	void(*XferErrorCallback)(struct __DMA_HandleTypeDef *hdma)	传输错误回调
读写	指针函数	void(*XferAbortCallback)(struct __DMA_HandleTypeDef *hdma)	传输中断回调
读写	uint32_t	ErrorCode	错误代码

读写	DMAMUX_TypeDef	DMAmuxInstance	DMAMUX 寄存器地址
----	----------------	----------------	-----------------

3.13.1 HAL_DMA_Init

3.13.1.1 功能介绍

初始化 DMA。

3.13.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_Init (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.2 HAL_DMA_DeInit

3.13.2.1 功能介绍

清除 DMA 初始化参数。

3.13.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_DeInit (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.3 HAL_DMA_Start

3.13.3.1 功能介绍

开启 DMA 转换。

3.13.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_Start (DMA_HandleTypeDef * hdma, uint32_t LSIAddress, uint32_t DstAddress, uint32_t DataLength)
输入	DMA_HandleTypeDef * hdma uint32_t LSIAddress uint32_t DstAddress uint32_t DataLength
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.4 HAL_DMA_Start_IT

3.13.4.1 功能介绍

以中断的方式，开启 DMA 转换。

3.13.4.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_Start_IT (DMA_HandleTypeDef * hdma, uint32_t LSIAddress, uint32_t DstAddress, uint32_t DataLength)
输入	DMA_HandleTypeDef * hdma uint32_t LSIAddress uint32_t DstAddress uint32_t DataLength
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.5 HAL_DMA_Abort

3.13.5.1 功能介绍

中止 DMA 转换。

3.13.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_Abort (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.6 HAL_DMA_Abort_IT

3.13.6.1 功能介绍

以中断的方式，中止 DMA 转换。

3.13.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_Abort_IT (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.7 HAL_DMA_PollForTransfer

3.13.7.1 功能介绍

DMA 传输轮询完成。

3.13.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_PollForTransfer (DMA_HandleTypeDef * hdma, HAL_DMA_LevelCompleteTypeDef CompleteLevel, uint32_t Timeout)
输入	DMA_HandleTypeDef * hdma HAL_DMA_LevelCompleteTypeDef CompleteLevel uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.8 HAL_DMA_IRQHandler

3.13.8.1 功能介绍

处理 DMA 中断请求。

3.13.8.2 接口定义

函数接口	void HAL_DMA_IRQHandler (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	无
资源使用	
说明	

3.13.9 HAL_DMA_RegisterCallback

3.13.9.1 功能介绍

注册回调。

3.13.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_RegisterCallback (DMA_HandleTypeDef * hdma, HAL_DMA_CallbackIDTypeDef CallbackID, void(*)(DMA_HandleTypeDef * _hdma) pCallback)
输入	DMA_HandleTypeDef * hdma HAL_DMA_CallbackIDTypeDef CallbackID void(*)(DMA_HandleTypeDef * _hdma) pCallback)
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.10 HAL_DMA_UnRegisterCallback

3.13.10.1 功能介绍

取消注册回调。

3.13.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMA_UnRegisterCallback (DMA_HandleTypeDef * hdma, HAL_DMA_CallbackIDTypeDef CallbackID)
输入	DMA_HandleTypeDef * hdma HAL_DMA_CallbackIDTypeDef CallbackID
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.11 HAL_DMA_GetState

3.13.11.1 功能介绍

获取 DMA 状态。

3.13.11.2 接口定义

函数接口	HAL_DMA_StateTypeDef HAL_DMA_GetState (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_DMA_StateTypeDef
资源使用	
说明	

3.13.12 HAL_DMA_GetError

3.13.12.1 功能介绍

获取 DMA 错误。

3.13.12.2 接口定义

函数接口	uint32_t HAL_DMA_GetError (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	错误值
资源使用	
说明	

3.13.13 HAL_DMAEx_ConfigMuxRequestGenerator

3.13.13.1 功能介绍

配置 DMA 通道使用的 DMAMUX 请求生成器。

3.13.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMAEx_ConfigMuxRequestGenerator (DMA_HandleTypeDef * hdma, HAL_DMA_MuxRequestGeneratorConfigTypeDef * pRequestGeneratorConfig)
------	--

输入	DMA_HandleTypeDef * hdma HAL_DMA_MuxRequestGeneratorConfigTypeDef * pRequestGeneratorConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.14 HAL_DMAEx_EnableMuxRequestGenerator

3.13.14.1 功能介绍

使能 DMA 通道使用的 DMAMUX 请求生成器。

3.13.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMAEx_EnableMuxRequestGenerator (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.15 HAL_DMAEx_DisableMuxRequestGenerator

3.13.15.1 功能介绍

禁止使能 DMA 通道使用的 DMAMUX 请求生成器。

3.13.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMAEx_DisableMuxRequestGenerator (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.13.16 HAL_DMAEx_ConfigMux

3.13.16.1 功能介绍

配置 DMA 通道的 DMAMUX 参数。

3.13.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_DMAEx_ConfigMux (DMA_HandleTypeDef * hdma, HAL_DMA_MuxConfigTypeDef * pConfig)
输入	DMA_HandleTypeDef * hdma HAL_DMA_MuxConfigTypeDef * pConfig
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.13.17 HAL_DMAEx_MUX_IRQHandler

3.13.17.1 功能介绍

处理 DMAMUX 中断请求。

3.13.17.2 接口定义

函数接口	void HAL_DMAEx_MUX_IRQHandler (DMA_HandleTypeDef * hdma)
输入	DMA_HandleTypeDef * hdma
输出	无
返回值	无
资源使用	
说明	

3.14 HDIV 模块

宏定义	数值	含义
HAL_HDIV_StateTypeDef		
HAL_HDIV_STATE_RESET	0x00	重置
HAL_HDIV_STATE_READY	0x01	就绪
HAL_HDIV_STATE_BUSY	0x02	忙
HAL_HDIV_TIMEOUT	0x03	超时
HAL_HDIV_STATE_ERROR	0x04	错误

属性	类型	字段名	含义
HDIV_TypeDef			
读写	uint32_t	DIVED	被除数寄存器
读写	uint32_t	DIV	除数器
只读	uint32_t	QUOT	商寄存器
只读	uint32_t	RMD	余数寄存器
读写	uint32_t	SIGN	符号寄存器
只读	uint32_t	STATUS	状态寄存器
HDIV_InitTypeDef			
读写	uint32_t	Sign	符号长度
CRC_HandleTypeDef			
读写	HDIV_TypeDef	Instance	寄存器基础地址
读写	HDIV_InitTypeDef	Init	初始化参数
读写	HAL_StatusTypeDef	Status	状态

读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_HDIV_StateTypeDef	State	HDIV 状态

3.14.1 HAL_HDIV_Init

3.14.1.1 功能介绍

初始化 HDIV。

3.14.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_HDIV_Init (HDIV_HandleTypeDef * hhdiv)
输入	HDIV_HandleTypeDef * hhdiv
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.14.2 HAL_HDIV_DeInit

3.14.2.1 功能介绍

清除 HDIV 初始化参数。

3.14.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_HDIV_DeInit (HDIV_HandleTypeDef * hhdiv)
输入	HDIV_HandleTypeDef * hhdiv
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.14.3 HAL_HDIV_MspInit

3.14.3.1 功能介绍

初始化 HDIV MSP。

3.14.3.2 接口定义

函数接口	void HAL_HDIV_MspInit (HDIV_HandleTypeDef * hhdiv)
输入	HDIV_HandleTypeDef * hhdiv
输出	无
返回值	无
资源使用	
说明	

3.14.4 HAL_HDIV_MspDeInit

3.14.4.1 功能介绍

清除 HDIV MSP 初始化参数。

3.14.4.2 接口定义

函数接口	void HAL_HDIV_MspDeInit (HDIV_HandleTypeDef * hdiv)
输入	HDIV_HandleTypeDef * hdiv
输出	无
返回值	无
资源使用	
说明	

3.14.5 HAL_HDIV_Calculate

3.14.5.1 功能介绍

启动 HDIV 操作。

3.14.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_HDIV_Calculate (HDIV_HandleTypeDef * hdiv, uint32_t dividend, uint32_t divisor, uint32_t * quotient, uint32_t * remainder)
输入	HDIV_HandleTypeDef * hdiv uint32_t dividend: [0x00000000, FFFFFFFF] uint32_t divisor: [0x00000000, FFFFFFFF] uint32_t * quotient: [0x00000000, FFFFFFFF] uint32_t * remainder: [0x00000000, FFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

3.14.6 HAL_HDIV_GetState

3.14.6.1 功能介绍

获取 HDIV 状态。

3.14.6.2 接口定义

函数接口	HAL_HDIV_StateTypeDef HAL_HDIV_GetState (HDIV_HandleTypeDef * hdiv)
输入	HDIV_HandleTypeDef * hdiv
输出	无
返回值	HAL_HDIV_StateTypeDef
资源使用	
说明	

3.15 I2C 模块

宏定义	数值	含义
HAL_I2C_StateTypeDef		
HAL_I2C_STATE_RESET	0x00	复位
HAL_I2C_STATE_READY	0x20	就绪
HAL_I2C_STATE_BUSY	0x24	忙
HAL_I2C_STATE_BUSY_TX	0x21	发送忙
HAL_I2C_STATE_BUSY_RX	0x22	接收忙
HAL_I2C_STATE_LISTEN	0x28	监听
HAL_I2C_STATE_BUSY_TX_LISTEN	0x29	监听发送忙
HAL_I2C_STATE_BUSY_RX_LISTEN	0x2A	监听接收忙
HAL_I2C_STATE_ABORT	0x60	中止
HAL_I2C_STATE_TIMEOUT	0xA0	超时
HAL_I2C_STATE_ERROR	0xE0	错误
HAL_I2C_ModeTypeDef		
HAL_I2C_MODE_NONE	0x00	无
HAL_I2C_MODE_MASTER	0x10	主模式
HAL_I2C_MODE_SLAVE	0x20	从模式
HAL_I2C_MODE_MEM	0x40	存储器模式

属性	类型	字段名	含义
I2C_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	OAR1	地址寄存器 1
读写	uint32_t	OAR2	地址寄存器 2
读写	uint32_t	TIMINGR	时钟配置寄存器
读写	uint32_t	ISR	中断和状态寄

			寄存器
只写	uint32_t	ICR	中断清除寄存器
只读	uint32_t	RXDR	接收数据寄存器
读写	uint32_t	TXDR	发送数据寄存器
I2C_InitTypeDef			
读写	uint32_t	Timing	时钟配置
读写	uint32_t	OwnAddress1	设备从模式地址1
读写	uint32_t	AddressingMode	地址模式
读写	uint32_t	DualAddressMode	双寻址模式
读写	uint32_t	OwnAddress2	设备从模式地址2
读写	uint32_t	OwnAddress2Masks	从模式地址2屏蔽
读写	uint32_t	GeneralCallMode	广播模式
读写	uint32_t	NoStretchMode	时钟低电平延长控制模式
I2C_HandleTypeDef			
读写	I2C_TypeDef	Instance	寄存器基础地

			址
读写	I2C_InitTypeDef	Init	初始化参数
读写	uint8_t	pBuffPtr	传输缓冲区的指针
读写	uint16_t	XferSize	传输大小
读写	uint16_t	XferCount	传输计数器
读写	uint32_t	XferOptions	传输选项
读写	uint32_t	PreviousState	通信前状态
读写	指针函数	HAL_StatusTypeDef(*XferISR)(struct __I2C_HandleTypeDef *hi2c, uint32_t ITFlags, uint32_t ITSources)	传递 IRQ 处理函数指针
读写	DMA_HandleTypeDef	hdmatx	DMA 句柄
读写	DMA_HandleTypeDef	hdmarx	DMA 句柄
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_I2C_StateTypeDef	State	I2C 状态
读写	HAL_I2C_ModeTypeDef	Mode	I2C 模式
读写	uint32_t	ErrorCode	错误代码
读写	uint32_t	AddrEventCount	事件计数器
读写	指针函数	void(*MasterTxCpltCallback)(struct __I2C_HandleTypeDef *hi2c)	主模式发送完成回调
读写	指针函数	void(*MasterRxCpltCallback)(struct __I2C_HandleTypeDef *hi2c);	主模式接收完成回调

读写	指针函数	<code>void(*SlaveTxCpltCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	从模式发送完成回调
读写	指针函数	<code>void(*SlaveRxCpltCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	从模式接收完成回调
读写	指针函数	<code>void(*ListenCpltCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	监听完成回调
读写	指针函数	<code>void(*MemTxCpltCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	存储器发送完成回调
读写	指针函数	<code>void(*MemRxCpltCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	存储器接收完成回调
读写	指针函数	<code>void(*ErrorCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	错误回调
读写	指针函数	<code>void(*AbortCpltCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	中止完成回调
读写	指针函数	<code>void(*AddrCallback)(struct __I2C_HandleTypeDef *hi2c, uint8_t TransferDirection,uint16_tAddrMatchCode);</code>	地址匹配完成回调
读写	指针函数	<code>void(*MspInitCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	MSP 初始化回调
读写	指针函数	<code>void(*MspDeInitCallback)(struct __I2C_HandleTypeDef *hi2c)</code>	MSP 非初始化回调

3.15.1 HAL_I2C_Init

3.15.1.1 功能介绍

初始化 I2C。

3.15.1.2 接口定义

函数接口	<code>HAL_StatusTypeDef HAL_I2C_Init (I2C_HandleTypeDef * hi2c)</code>
输入	<code>I2C_HandleTypeDef * hi2c</code>
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.2 HAL_I2C_DeInit

3.15.2.1 功能介绍

清除 I2C 初始化参数。

3.15.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_DeInit (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.3 HAL_I2C_MspInit

3.15.3.1 功能介绍

初始化 I2C MSP。

3.15.3.2 接口定义

函数接口	void HAL_I2C_MspInit (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.4 HAL_I2C_MspDeInit

3.15.4.1 功能介绍

清除 I2C MSP 初始化参数。

3.15.4.2 接口定义

函数接口	void HAL_I2C_MspDeInit (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.5 HAL_I2C_Master_Transmit

3.15.5.1 功能介绍

在主模式下以阻塞模式发送一定量的数据。

3.15.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Transmit (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.6 HAL_I2C_Master_Receive

3.15.6.1 功能介绍

在主模式下以阻塞模式接收一定量的数据。

3.15.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Receive (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.7 HAL_I2C_Slave_Transmit

3.15.7.1 功能介绍

在从模式下以阻塞模式发送一定量的数据。

3.15.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Transmit (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.15.8 HAL_I2C_Slave_Receive

3.15.8.1 功能介绍

在从模式下以阻塞模式接收一定量的数据。

3.15.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Receive (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.9 HAL_I2C_Mem_Write

3.15.9.1 功能介绍

以阻塞模式将一定数量的数据写入特定的内存地址。

3.15.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Mem_Write (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t MemAddress uint16_t MemAddSize uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.10 HAL_I2C_Mem_Read

3.15.10.1 功能介绍

以阻塞模式将一定数量的数据读取特定的内存地址。

3.15.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Mem_Read (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize,
------	---

	uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t MemAddress uint16_t MemAddSize uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.11 HAL_I2C_IsDeviceReady

3.15.11.1 功能介绍

检查目标设备是否准备好进行通信。

3.15.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_IsDeviceReady (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint32_t Trials, uint32_t Timeout)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t Trials uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.12 HAL_I2C_Master_Transmit_IT

3.15.12.1 功能介绍

在主模式下以中断的方式以非阻塞模式发送一定量的数据。

3.15.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Transmit_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.13 HAL_I2C_Master_Receive_IT

3.15.13.1 功能介绍

在主模式下以中断的方式以非阻塞模式接收一定量的数据。

3.15.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Receive_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.14 HAL_I2C_Slave_Transmit_IT

3.15.14.1 功能介绍

在从模式下以中断的方式以非阻塞模式发送一定量的数据。

3.15.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Transmit_IT (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.15 HAL_I2C_Slave_Receive_IT

3.15.15.1 功能介绍

在从模式下以中断的方式以非阻塞模式接收一定量的数据。

3.15.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Receive_IT (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.16 HAL_I2C_Mem_Write_IT

3.15.16.1 功能介绍

以中断的方式以非阻塞模式将一定数量的数据写入特定的内存地址。

3.15.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Mem_Write_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t MemAddress uint16_t MemAddSize uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.17 HAL_I2C_Mem_Read_IT

3.15.17.1 功能介绍

以中断的方式以非阻塞模式将一定数量的数据读取特定的内存地址。

3.15.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Mem_Read_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t MemAddress uint16_t MemAddSize uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.18 HAL_I2C_Master_Seq_Transmit_IT

3.15.18.1 功能介绍

在主模式下以中断的方式以非阻塞模式顺序发送一定量的数据。

3.15.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Seq_Transmit_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData,
------	--

	uint16_t Size, uint32_t XferOptions)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.19 HAL_I2C_Master_Seq_Receive_IT

3.15.19.1 功能介绍

在主模式下以中断的方式以非阻塞模式顺序接收一定量的数据。

3.15.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Seq_Receive_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.20 HAL_I2C_Slave_Seq_Transmit_IT

3.15.20.1 功能介绍

在从模式下以中断的方式以非阻塞模式顺序发送一定量的数据。

3.15.20.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Seq_Transmit_IT (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
------	--

输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.21 HAL_I2C_Slave_Seq_Receive_IT

3.15.21.1 功能介绍

在从模式下以中断的方式以非阻塞模式顺序接收一定量的数据。

3.15.21.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Seq_Receive_IT (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.22 HAL_I2C_EnableListen_IT

3.15.22.1 功能介绍

使能中断地址监听模式。

3.15.22.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_EnableListen_IT (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.15.23 HAL_I2C_DisableListen_IT

3.15.23.1 功能介绍

禁止使能中断地址监听模式。

3.15.23.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_DisableListen_IT (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.24 HAL_I2C_Master_Abort_IT

3.15.24.1 功能介绍

中止 I2C 主模式 IT 或 DMA 进程的通信。

3.15.24.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Abort_IT (I2C_HandleTypeDef * hi2c, uint16_t DevAddress)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.25 HAL_I2C_Master_Transmit_DMA

3.15.25.1 功能介绍

在主模式下以 DMA 方式以非阻塞模式发送一定量的数据。

3.15.25.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Transmit_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.26 HAL_I2C_Master_Receive_DMA

3.15.26.1 功能介绍

在主模式下以 DMA 方式以非阻塞模式接收一定量的数据。

3.15.26.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Receive_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.27 HAL_I2C_Slave_Transmit_DMA

3.15.27.1 功能介绍

在从模式下以 DMA 方式以非阻塞模式发送一定量的数据。

3.15.27.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Transmit_DMA (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.28 HAL_I2C_Slave_Receive_DMA

3.15.28.1 功能介绍

在从模式下以 DMA 方式以非阻塞模式接收一定量的数据。

3.15.28.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Receive_DMA (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.15.29 HAL_I2C_Mem_Write_DMA

3.15.29.1 功能介绍

以 DMA 方式以非阻塞模式将一定数量的数据写入特定的内存地址。

3.15.29.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Mem_Write_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t MemAddress uint16_t MemAddSize uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.30 HAL_I2C_Mem_Read_DMA

3.15.30.1 功能介绍

以 DMA 方式以非阻塞模式将一定数量的数据读取特定的内存地址。

3.15.30.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Mem_Read_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint16_t MemAddress, uint16_t MemAddSize, uint8_t * pData, uint16_t Size)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint16_t MemAddress uint16_t MemAddSize uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.31 HAL_I2C_Master_Seq_Transmit_DMA

3.15.31.1 功能介绍

在主模式下以 DMA 方式以非阻塞模式顺序发送一定量的数据。

3.15.31.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Seq_Transmit_DMA
------	---

	(I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.32 HAL_I2C_Master_Seq_Receive_DMA

3.15.32.1 功能介绍

在主模式下以 DMA 方式以非阻塞模式顺序接收一定量的数据。

3.15.32.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Master_Seq_Receive_DMA (I2C_HandleTypeDef * hi2c, uint16_t DevAddress, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
输入	I2C_HandleTypeDef * hi2c uint16_t DevAddress uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.33 HAL_I2C_Slave_Seq_Transmit_DMA

3.15.33.1 功能介绍

在从模式下以 DMA 方式以非阻塞模式顺序发送一定量的数据。

3.15.33.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Seq_Transmit_DMA (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
------	--

输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.34 HAL_I2C_Slave_Seq_Receive_DMA

3.15.34.1 功能介绍

在从模式下以 DMA 方式以非阻塞模式顺序接收一定量的数据。

3.15.34.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2C_Slave_Seq_Receive_DMA (I2C_HandleTypeDef * hi2c, uint8_t * pData, uint16_t Size, uint32_t XferOptions)
输入	I2C_HandleTypeDef * hi2c uint8_t * pData uint16_t Size uint32_t XferOptions: { I2C_FIRST_FRAME, I2C_FIRST_AND_NEXT_FRAME, I2C_NEXT_FRAME, I2C_FIRST_AND_LAST_FRAME, I2C_LAST_FRAME, I2C_LAST_FRAME_NO_STOP, I2C_OTHER_FRAME, I2C_OTHER_AND_LAST_FRAME }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.35 HAL_I2C_EV_IRQHandler

3.15.35.1 功能介绍

处理 I2C 事件中断请求。

3.15.35.2 接口定义

函数接口	void HAL_I2C_EV_IRQHandler (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.36 HAL_I2C_ER_IRQHandler

3.15.36.1 功能介绍

处理 I2C 事件错误中断请求。

3.15.36.2 接口定义

函数接口	void HAL_I2C_ER_IRQHandler (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.37 HAL_I2C_MasterTxCpltCallback

3.15.37.1 功能介绍

主模式发送传输完成回调。

3.15.37.2 接口定义

函数接口	void HAL_I2C_MasterTxCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.38 HAL_I2C_MasterRxCpltCallback

3.15.38.1 功能介绍

主模式接收传输完成回调。

3.15.38.2 接口定义

函数接口	void HAL_I2C_MasterRxCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.39 HAL_I2C_SlaveTxCpltCallback

3.15.39.1 功能介绍

从模式发送传输完成回调。

3.15.39.2 接口定义

函数接口	void HAL_I2C_SlaveTxCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无

返回值	无
资源使用	
说明	

3.15.40 HAL_I2C_SlaveRxCpltCallback

3.15.40.1 功能介绍

从模式接收传输完成回调。

3.15.40.2 接口定义

函数接口	void HAL_I2C_SlaveRxCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.41 HAL_I2C_AddrCallback

3.15.41.1 功能介绍

从模式地址匹配回调。

3.15.41.2 接口定义

函数接口	void HAL_I2C_AddrCallback (I2C_HandleTypeDef * hi2c, uint8_t TransferDirection, uint16_t AddrMatchCode)
输入	I2C_HandleTypeDef * hi2c uint8_t TransferDirection uint16_t AddrMatchCode
输出	无
返回值	无
资源使用	
说明	

3.15.42 HAL_I2C_ListenCpltCallback

3.15.42.1 功能介绍

监听完成回调。

3.15.42.2 接口定义

函数接口	void HAL_I2C_ListenCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.43 HAL_I2C_MemTxCpltCallback

3.15.43.1 功能介绍

存储器发送传输完成回调。

3.15.43.2 接口定义

函数接口	void HAL_I2C_MemTxCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.44 HAL_I2C_MemRxCpltCallback

3.15.44.1 功能介绍

存储器接收传输完成回调。

3.15.44.2 接口定义

函数接口	void HAL_I2C_MemRxCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.45 HAL_I2C_ErrorCallback

3.15.45.1 功能介绍

I2C 错误回调。

3.15.45.2 接口定义

函数接口	void HAL_I2C_ErrorCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	无
资源使用	
说明	

3.15.46 HAL_I2C_AbortCpltCallback

3.15.46.1 功能介绍

I2C 中止回调。

3.15.46.2 接口定义

函数接口	void HAL_I2C_AbortCpltCallback (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无

返回值	无
资源使用	
说明	

3.15.47 HAL_I2C_GetState

3.15.47.1 功能介绍

获取 I2C 状态。

3.15.47.2 接口定义

函数接口	HAL_I2C_StateTypeDef HAL_I2C_GetState (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_I2C_StateTypeDef
资源使用	
说明	

3.15.48 HAL_I2C_GetMode

3.15.48.1 功能介绍

获取 I2C 模式。

3.15.48.2 接口定义

函数接口	HAL_I2C_ModeTypeDef HAL_I2C_GetMode (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_I2C_ModeTypeDef
资源使用	
说明	

3.15.49 HAL_I2C_GetError

3.15.49.1 功能介绍

获取 I2C 错误。

3.15.49.2 接口定义

函数接口	uint32_t HAL_I2C_GetError (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	错误值
资源使用	
说明	

3.15.50 HAL_I2CEx_ConfigDigitalFilter

3.15.50.1 功能介绍

配置 I2C 数字噪声滤波器。

3.15.50.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2CEx_ConfigDigitalFilter (I2C_HandleTypeDef * hi2c, uint32_t DigitalFilter)
输入	I2C_HandleTypeDef * hi2c uint32_t DigitalFilter: [0x00, 0x0F]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.51 HAL_I2CEx_EnableWakeUp

3.15.51.1 功能介绍

使能 I2C 从停止模式唤醒。

3.15.51.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2CEx_EnableWakeUp (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.15.52 HAL_I2CEx_DisableWakeUp

3.15.52.1 功能介绍

禁止使能 I2C 从停止模式唤醒。

3.15.52.2 接口定义

函数接口	HAL_StatusTypeDef HAL_I2CEx_DisableWakeUp (I2C_HandleTypeDef * hi2c)
输入	I2C_HandleTypeDef * hi2c
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16 IRDA 模块

属性	类型	字段名	含义
USART_TypeDef			
读写	uint32_t	CR1	控制寄存器 1

读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	CR3	控制寄存器 3
读写	uint32_t	BRR	波特率分频寄存器
读写	uint32_t	GTPR	保护时间和预分频器寄存器
读写	uint32_t	RTOR	超时及块传输长度寄存器
只写	uint32_t	RQR	请求寄存器
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断标志清零寄存器
只读	uint32_t	RDR	接收数据寄存器
读写	uint32_t	TDR	发送数据寄存器
读写	uint32_t	PRESC	预分频器寄存器
IRDA_InitTypeDef			
读写	uint32_t	BaudRate	波特率
读写	uint32_t	WordLength	字符长度
读写	uint32_t	Parity	RS485 收发器使能信号极性
读写	uint32_t	Mode	模式
读写	uint32_t	Prescaler	分频值
读写	uint32_t	ClockPrescaler	时钟分频值

IRDA_HandleTypeDef			
读写	USART_TypeDef	Instance	寄存器地址
读写	IRDA_InitTypeDef	Init	初始化参数
读写	uint8_t	pTxBuffPtr	指向 IRDA Tx 传输缓冲区的指针
读写	uint16_t	TxXferSize	发送传输大小
读写	uint16_t	TxXferCount	发送传输计数器
读写	uint8_t	pRxBuffPtr	指向 IRDA Rx 传输缓冲区的指针
读写	uint16_t	RxXferSize	接收传输大小
读写	uint16_t	RxXferCount	接收传输计数器
读写	uint16_t	Mask	RX RDR 寄存器掩码
读写	DMA_HandleTypeDef	hdmatx	DMA 句柄
读写	DMA_HandleTypeDef	hdmarx	DMA 句柄
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_IRDA_StateTypeDef	gState	IRDA 状态
读写	HAL_IRDA_StateTypeDef	RxState	IRDA 状态
读写	uint32_t	ErrorCode	错误代码
读写	指针函数	void(*TxHalfCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	发送传输

			完成一半回调
读写	指针函数	void(*TxCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	发送传输完成回调
读写	指针函数	void(*RxHalfCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	接收传输完成一半回调
读写	指针函数	void(*RxCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	接收传输完成回调
读写	指针函数	void(*ErrorCallback)(struct __IRDA_HandleTypeDef *hirda)	错误回调
读写	指针函数	void(*AbortCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	中止完成回调
读写	指针函数	void(*AbortTransmitCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	中止发送完成回调
读写	指针函数	void(*AbortReceiveCpltCallback)(struct __IRDA_HandleTypeDef *hirda)	中止接收完成回调
读写	指针函数	void(*MspInitCallback)(struct __IRDA_HandleTypeDef *hirda)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback)(struct __IRDA_HandleTypeDef *hirda)	MSP 非初始化回调

3.16.1 HAL_IRDA_Init

3.16.1.1 功能介绍

IRDA 初始化。

3.16.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Init (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.2 HAL_IRDA_DeInit

3.16.2.1 功能介绍

清除 IRDA 初始化参数。

3.16.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_DeInit (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.3 HAL_IRDA_MspInit

3.16.3.1 功能介绍

IRDA MSP 初始化。

3.16.3.2 接口定义

函数接口	void HAL_IRDA_MspInit (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.4 HAL_IRDA_DeInit

3.16.4.1 功能介绍

清除 IRDA MSP 初始化参数。

3.16.4.2 接口定义

函数接口	void HAL_IRDA_MspDeInit (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.5 HAL_IRDA_Transmit

3.16.5.1 功能介绍

以阻塞模式发送数据。

3.16.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Transmit (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	IRDA_HandleTypeDef * hirda uint8_t * pData uint16_t Size uint32_t Timeout

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.6 HAL_IRDA_Receive

3.16.6.1 功能介绍

以阻塞模式接收数据。

3.16.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Receive (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	IRDA_HandleTypeDef * hirda uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.7 HAL_IRDA_Transmit_IT

3.16.7.1 功能介绍

以中断模式发送数据。

3.16.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Transmit_IT (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
输入	IRDA_HandleTypeDef * hirda uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.8 HAL_IRDA_Receive_IT

3.16.8.1 功能介绍

以中断模式接收数据。

3.16.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Receive_IT (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
输入	IRDA_HandleTypeDef * hirda uint8_t * pData uint16_t Size

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.9 HAL_IRDA_Transmit_DMA

3.16.9.1 功能介绍

以 DMA 模式发送数据。

3.16.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Transmit_DMA (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
输入	IRDA_HandleTypeDef * hirda uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.10 HAL_IRDA_Receive_DMA

3.16.10.1 功能介绍

以 DMA 模式接收数据。

3.16.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Receive_DMA (IRDA_HandleTypeDef * hirda, uint8_t * pData, uint16_t Size)
输入	IRDA_HandleTypeDef * hirda uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.11 HAL_IRDA_DMABase

3.16.11.1 功能介绍

暂停 DMA 传输数据。

3.16.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_DMABase (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.16.12 HAL_IRDA_DMAResume

3.16.12.1 功能介绍

恢复 DMA 传输数据。

3.16.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_DMAResume (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.13 HAL_IRDA_DMAStop

3.16.13.1 功能介绍

停止 DMA 传输数据。

3.16.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_DMAStop (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.14 HAL_IRDA_Abort

3.16.14.1 功能介绍

在阻塞模式下，中止正在进行的传输。

3.16.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Abort (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.15 HAL_IRDA_AbortTransmit

3.16.15.1 功能介绍

在阻塞模式下，中止正在进行的发送传输。

3.16.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_AbortTransmit (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.16 HAL_IRDA_AbortReceive

3.16.16.1 功能介绍

在阻塞模式下，中止正在进行的接收传输。

3.16.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_AbortReceive (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.17 HAL_IRDA_Abort_IT

3.16.17.1 功能介绍

在中断模式下，中止正在进行的传输。

3.16.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_Abort_IT (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.18 HAL_IRDA_AbortTransmit_IT

3.16.18.1 功能介绍

在中断模式下，中止正在进行的发送传输。

3.16.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_AbortTransmit_IT (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.19 HAL_IRDA_AbortReceive_IT

3.16.19.1 功能介绍

在中断模式下，中止正在进行的接收传输。

3.16.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_IRDA_AbortReceive_IT (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.16.20 HAL_IRDA_IRQHandler

3.16.20.1 功能介绍

处理 IRDA 中断请求。

3.16.20.2 接口定义

函数接口	void HAL_IRDA_IRQHandler (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.21 HAL_IRDA_TxCpltCallback

3.16.21.1 功能介绍

发送传输完成回调。

3.16.21.2 接口定义

函数接口	void HAL_IRDA_TxCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.22 HAL_IRDA_RxCpltCallback

3.16.22.1 功能介绍

接收传输完成回调。

3.16.22.2 接口定义

函数接口	void HAL_IRDA_RxCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.23 HAL_IRDA_TxHalfCpltCallback

3.16.23.1 功能介绍

发送传输完成一半回调。

3.16.23.2 接口定义

函数接口	void HAL_IRDA_TxHalfCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.24 HAL_IRDA_RxHalfCpltCallback

3.16.24.1 功能介绍

接收传输完成一半回调。

3.16.24.2 接口定义

函数接口	void HAL_IRDA_RxHalfCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.25 HAL_IRDA_ErrorCallback

3.16.25.1 功能介绍

错误回调。

3.16.25.2 接口定义

函数接口	void HAL_IRDA_ErrorCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.26 HAL_IRDA_AbortCpltCallback

3.16.26.1 功能介绍

IRDA 中止完成回调。

3.16.26.2 接口定义

函数接口	void HAL_IRDA_AbortCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.27 HAL_IRDA_AbortTransmitCpltCallback

3.16.27.1 功能介绍

IRDA 中止发送完成回调。

3.16.27.2 接口定义

函数接口	void HAL_IRDA_AbortTransmitCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.28 HAL_IRDA_AbortReceiveCpltCallback

3.16.28.1 功能介绍

IRDA 中止接收完成回调。

3.16.28.2 接口定义

函数接口	void HAL_IRDA_AbortReceiveCpltCallback (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	无
资源使用	
说明	

3.16.29 HAL_IRDA_GetState

3.16.29.1 功能介绍

获取 IRDA 状态。

3.16.29.2 接口定义

函数接口	HAL_IRDA_StateTypeDef HAL_IRDA_GetState (IRDA_HandleTypeDef * hirda)
------	--

输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	HAL_IRDA_StateTypeDef
资源使用	
说明	

3.16.30 HAL_IRDA_GetError

3.16.30.1 功能介绍

获取 IRDA 错误。

3.16.30.2 接口定义

函数接口	uint32_t HAL_IRDA_GetError (IRDA_HandleTypeDef * hirda)
输入	IRDA_HandleTypeDef * hirda
输出	无
返回值	Error Code
资源使用	
说明	

3.17 LCDLED 模块

宏定义	数值	含义
HAL_LCDLED_StateTypeDef		
HAL_LCDLED_STATE_RESET	0x00	重置
HAL_LCDLED_STATE_READY	0x01	就绪
HAL_LCDLED_STATE_BUSY	0x02	忙
HAL_LCDLED_STATE_TIMEOUT	0x03	超时
HAL_LCDLED_STATE_ERROR	0x04	错误

属性	类型	字段名	含义
LCDLED_TypeDef			
读写	uint32_t	CR	控制寄存器
读写	uint32_t	RAM0	显示存储器
读写	uint32_t	RAM1	显示存储器
读写	uint32_t	RAM2	显示存储器
读写	uint32_t	RAM3	显示存储器
读写	uint32_t	RAM4	显示存储器
读写	uint32_t	RAM5	显示存储器
读写	uint32_t	RAM6	显示存储器
读写	uint32_t	RAM7	显示存储器

LCDLED_InitTypeDef			
读写	uint32_t	Mode	模式
读写	uint32_t	LCDClockSource	LCD 时钟源
读写	uint32_t	BrightLevel	亮度等级
读写	uint32_t	Prescaler	分频值
读写	uint32_t	LEDBrightMode	亮度模式
读写	uint32_t	Duty	占空比
LCDLED_HandleTypeDef			
读写	LCDLED_TypeDef	Instance	寄存器基础地址
读写	LCDLED_InitTypeDef	Init	初始化参数
读写	HAL_StatusTypeDef	Status	状态
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_LCDLED_StateTypeDef	State	LCDLED 状态

3.17.1 HAL_LCDLED_Init

3.17.1.1 功能介绍

初始化 LCD\LED。

3.17.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LCDLED_Init (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.17.2 HAL_LCDLED_DeInit

3.17.2.1 功能介绍

清除 LCD\LED 初始化参数。

3.17.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LCDLED_DeInit (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.17.3 HAL_LCDLED_MspInit

3.17.3.1 功能介绍

初始化 LCD\LED MSP。

3.17.3.2 接口定义

函数接口	void HAL_LCDLED_MspInit (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	无
资源使用	
说明	

3.17.4 HAL_LCDLED_MspDeInit

3.17.4.1 功能介绍

清除 LCD\LED MSP 初始化参数。

3.17.4.2 接口定义

函数接口	void HAL_LCDLED_MspDeInit (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	无
资源使用	
说明	

3.17.5 HAL_LCDLED_Display

3.17.5.1 功能介绍

启动 LCD/LED 显示生成。

3.17.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LCDLED_Display (LCDLED_HandleTypeDef * hlcdled, uint32_t displayNum, uint32_t * data)
输入	LCDLED_HandleTypeDef * hlcdled uint32_t displayNum: [0x1, 0x8] uint32_t * data: [0x00000000, 0xFFFFFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.17.6 HAL_LCDLED_Hold

3.17.6.1 功能介绍

保持 LCD/LED 显示。

3.17.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LCDLED_Hold (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.17.7 HAL_LCDLED_Scan

3.17.7.1 功能介绍

扫描 LCD/LED 显示。

3.17.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LCDLED_Scan (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.17.8 HAL_LCDLED_GetState

3.17.8.1 功能介绍

获取 LCD/LED 状态。

3.17.8.2 接口定义

函数接口	HAL_LCDLED_StateTypeDef HAL_LCDLED_GetState (LCDLED_HandleTypeDef * hlcdled)
输入	LCDLED_HandleTypeDef * hlcdled
输出	无
返回值	HAL_LCDLED_StateTypeDef
资源使用	
说明	

3.18 LPTIM 模块

宏定义	数值	含义
HAL_LPTIM_StateTypeDef		
HAL_LPTIM_STATE_RESET	0x00	复位
HAL_LPTIM_STATE_READY	0x01	就绪
HAL_LPTIM_STATE_BUSY	0x02	忙
HAL_LPTIM_STATE_TIMEOUT	0x03	超时

HAL_LPTIM_STATE_ERROR	0x04	错误
-----------------------	------	----

属性	类型	字段名	含义
LPTIM_TypeDef			
只读	uint32_t	ISR	中断状态寄存器
只写	uint32_t	ICR	中断清零寄存器
读写	uint32_t	IER	中断使能寄存器
读写	uint32_t	CFGR	配置寄存器
读写	uint32_t	CR	控制寄存器
读写	uint32_t	CMP	比较寄存器
读写	uint32_t	ARR	自动重载寄存器
只读	uint32_t	CNT	计数器寄存器
读写	uint32_t	CFGR2	配置寄存

			器 2
LPTIM_ClockConfigTypeDef			
读 写	uint32_t	Source	时 钟 源
读 写	uint32_t	Prescaler	分 频 值
LPTIM_ULPClockConfigTypeDef			
读 写	uint32_t	Polarity	极 性
读 写	uint32_t	SampleTime	采 样 时间
LPTIM_TriggerConfigTypeDef			
读 写	uint32_t	Source	触 发 源
读 写	uint32_t	ActiveEdge	触 发 沿
读 写	uint32_t	SampleTime	采 样 时间
LPTIM_InitTypeDef			
读 写	LPTIM_ClockConfigTypeDef	Clock	时 钟 参数
读 写	LPTIM_ULPClockConfigTypeDef	UltraLowPowerClock	超 低 功 耗 时 钟 参数
读 写	LPTIM_TriggerConfigTypeDef	Trigger	触 发 参数
读 写	uint32_t	OutputPolarity	输 出 极 性
读 写	uint32_t	UpdateMode	输 出 模 式
读 写	uint32_t	CounterSource	计 数 源
读 写	uint32_t	InputFilterPrescaler	输 入 滤 波

			分 频 值
读 写	uint32_t	Input1Polarity	输入 1 极性
读 写	uint32_t	Input2Polarity	输入 2 极性
读 写	uint32_t	Input1Source	输入 1 源
读 写	uint32_t	Input2Source	输入 2 源
读 写	uint32_t	DualMode	双 编 码 器 模式
LPTIM_HandleTypeDef			
读 写	LPTIM_TypeDef	Instance	寄 存 器 地 址
读 写	LPTIM_InitTypeDef	Init	初 始 化 参 数
读 写	HAL_StatusTypeDef	Status	锁定
读 写	HAL_LockTypeDef	Lock	状态
读 写	HAL_LPTIM_StateTypeDef	State	LPTI M 状 态
读 写	uint32_t	void(*MspInitCallback)(struct __LPTIM_HandleTypeDef *hlptim)	MSP 初 始 化 回 调
读 写	uint32_t	void(*MspDeInitCallback)(struct __LPTIM_HandleTypeDef *hlptim)	MSP 非 初 始 化 回 调

读 写	uint32_t	void(*CompareMatchCallback)(struct __LPTIM_HandleTypeDef *hlptim)	比 较 匹 配 回 调
读 写	uint32_t	void(*AutoReloadMatchCallback)(struct __LPTIM_HandleTypeDef *hlptim)	自 动 重 载 匹 配 回 调
读 写	uint32_t	void(*TriggerCallback)(struct __LPTIM_HandleTypeDef *hlptim)	外 部 触 发 回 调
读 写	uint32_t	void(*CompareWriteCallback)(struct __LPTIM_HandleTypeDef *hlptim)	比 较 寄 存 器 写 完 成 回 调
读 写	uint32_t	void(*AutoReloadWriteCallback)(struct __LPTIM_HandleTypeDef *hlptim)	自 动 重 新 加 载 寄 存 器 写 完 成 回 调
读 写	uint32_t	void(*DirectionUpCallback)(struct __LPTIM_HandleTypeDef *hlptim)	向 上 计 数 方 向 改 变 回 调
读 写	uint32_t	void(*DirectionDownCallback)(struct __LPTIM_HandleTypeDef *hlptim)	向 下 计 数 方 向 改 变 回 调
读 写	uint32_t	void(*DualInput1IdleErrorCallback)(struct __LPTIM_HandleTypeDef *hlptim)	双 编 码 模 式 下

			的 输 入 1 错 误 回 调
读 写	uint32_t	void(*DualInput2IdleErrorCallback)(struct __LPTIM_HandleTypeDef *hlptim)	双 编 码 模 式 下 的 输 入 2 错 误 回 调
读 写	uint32_t	void(*DualB2BErrorCallback)(struct __LPTIM_HandleTypeDef *hlptim)	在 双 编 码 模 式 下, 输 入 1 和 输入 2 没 有 空 格 错 误 回 调
读 写	uint32_t	void(*DualWaveFormErrorCallback)(struc t __LPTIM_HandleTypeDef *hlptim)	双 编 码 模 式 下 的 输 入 1 和 输入 2 波 形 错 误 回 调

3.18.1 HAL_LPTIM_Init

3.18.1.1 功能介绍

初始化 LPTIM。

3.18.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Init (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.2 HAL_LPTIM_DeInit

3.18.2.1 功能介绍

清除 LPTIM 初始化参数。

3.18.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_DeInit (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.3 HAL_LPTIM_MspInit

3.18.3.1 功能介绍

初始化 LPTIM MSP。

3.18.3.2 接口定义

函数接口	void HAL_LPTIM_MspInit (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.4 HAL_LPTIM_MspDeInit

3.18.4.1 功能介绍

清除 LPTIM MSP 初始化参数。

3.18.4.2 接口定义

函数接口	void HAL_LPTIM_MspDeInit (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	

说明	
----	--

3.18.5 HAL_LPTIM_PWM_Start

3.18.5.1 功能介绍

启动 LPTIM PWM 生成。

3.18.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_PWM_Start (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Pulse)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Pulse: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.6 HAL_LPTIM_PWM_Stop

3.18.6.1 功能介绍

停止 LPTIM PWM 生成。

3.18.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_PWM_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.7 HAL_LPTIM_PWM_Start_IT

3.18.7.1 功能介绍

以中断模式启动 LPTIM PWM 生成。

3.18.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_PWM_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Pulse)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Pulse: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.18.8 HAL_LPTIM_PWM_Stop_IT

3.18.8.1 功能介绍

以中断模式停止 LPTIM PWM 生成。

3.18.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_PWM_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.9 HAL_LPTIM_OnePulse_Start

3.18.9.1 功能介绍

启动 LPTIM 单脉冲生成。

3.18.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_OnePulse_Start (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Pulse)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Pulse: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.10 HAL_LPTIM_OnePulse_Stop

3.18.10.1 功能介绍

停止 LPTIM 单脉冲生成。

3.18.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_OnePulse_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.11 HAL_LPTIM_OnePulse_Start_IT

3.18.11.1 功能介绍

以中断模式启动 LPTIM 单脉冲生成。

3.18.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_OnePulse_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Pulse)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Pulse: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.12 HAL_LPTIM_OnePulse_Stop_IT

3.18.12.1 功能介绍

以中断方式停止 LPTIM 单脉冲生成。

3.18.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_OnePulse_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.13 HAL_LPTIM_SetOnce_Start

3.18.13.1 功能介绍

以单触发模式启动 LPTIM。

3.18.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Start (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Pulse)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Pulse: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.18.14 HAL_LPTIM_SetOnce_Stop

3.18.14.1 功能介绍

以单触发模式停止 LPTIM。

3.18.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.15 HAL_LPTIM_SetOnce_Start_IT

3.18.15.1 功能介绍

在中断模式下以单触发模式启动 LPTIM。

3.18.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Pulse)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Pulse: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.16 HAL_LPTIM_SetOnce_Stop_IT

3.18.16.1 功能介绍

在中断模式下以单触发模式启动 LPTIM。

3.18.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_SetOnce_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.17 HAL_LPTIM_Encoder_Start

3.18.17.1 功能介绍

启动正交编码器接口。

3.18.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Encoder_Start (LPTIM_HandleTypeDef * hlptim, uint32_t Period)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.18 HAL_LPTIM_Encoder_Stop

3.18.18.1 功能介绍

停止正交编码器接口。

3.18.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Encoder_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.19 HAL_LPTIM_Encoder_Start_IT

3.18.19.1 功能介绍

以中断模式启动正交编码器接口。

3.18.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Encoder_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.20 HAL_LPTIM_Encoder_Stop_IT

3.18.20.1 功能介绍

以中断模式停止正交编码器接口。

3.18.20.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Encoder_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.21 HAL_LPTIM_DualEncoder_Start

3.18.21.1 功能介绍

启动非交编码器接口。

3.18.21.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_DualEncoder_Start (LPTIM_HandleTypeDef * hlptim, uint32_t Period)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.22 HAL_LPTIM_DualEncoder_Stop

3.18.22.1 功能介绍

停止非交编码器接口。

3.18.22.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_DualEncoder_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.23 HAL_LPTIM_DualEncoder_Start_IT

3.18.23.1 功能介绍

以中断模式启动非交编码器接口。

3.18.23.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_DualEncoder_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.24 HAL_LPTIM_DualEncoder_Stop_IT

3.18.24.1 功能介绍

以中断模式停止非交编码器接口。

3.18.24.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_DualEncoder_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.25 HAL_LPTIM_TimeOut_Start

3.18.25.1 功能介绍

启动超时功能。

3.18.25.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_TimeOut_Start (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Timeout)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Timeout: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.26 HAL_LPTIM_TimeOut_Stop

3.18.26.1 功能介绍

停止超时功能。

3.18.26.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_TimeOut_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.27 HAL_LPTIM_TimeOut_Start_IT

3.18.27.1 功能介绍

以中断模式启动超时功能。

3.18.27.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_TimeOut_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period, uint32_t Timeout)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF] uint32_t Timeout: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.28 HAL_LPTIM_TimeOut_Stop_IT

3.18.28.1 功能介绍

以中断模式停止超时功能。

3.18.28.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_TimeOut_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.29 HAL_LPTIM_Counter_Start

3.18.29.1 功能介绍

启动计数功能。

3.18.29.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Counter_Start
------	---

	(LPTIM_HandleTypeDef * hlptim, uint32_t Period)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.30 HAL_LPTIM_Counter_Stop

3.18.30.1 功能介绍

停止计数功能。

3.18.30.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Counter_Stop (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.31 HAL_LPTIM_Counter_Start_IT

3.18.31.1 功能介绍

以中断模式启动计数功能。

3.18.31.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Counter_Start_IT (LPTIM_HandleTypeDef * hlptim, uint32_t Period)
输入	LPTIM_HandleTypeDef * hlptim uint32_t Period: [0x0000, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.32 HAL_LPTIM_Counter_Stop_IT

3.18.32.1 功能介绍

以中断模式停止计数功能。

3.18.32.2 接口定义

函数接口	HAL_StatusTypeDef HAL_LPTIM_Counter_Stop_IT (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.18.33 HAL_LPTIM_ReadCounter

3.18.33.1 功能介绍

读取当前计数值。

3.18.33.2 接口定义

函数接口	uint32_t HAL_LPTIM_ReadCounter (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	计数值
资源使用	
说明	

3.18.34 HAL_LPTIM_ReadAutoReload

3.18.34.1 功能介绍

读取当前的自动重新加载值。

3.18.34.2 接口定义

函数接口	uint32_t HAL_LPTIM_ReadAutoReload (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	自动重新加载值
资源使用	
说明	

3.18.35 HAL_LPTIM_ReadCompare

3.18.35.1 功能介绍

读取当前的比较值。

3.18.35.2 接口定义

函数接口	uint32_t HAL_LPTIM_ReadCompare (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	比较值
资源使用	
说明	

3.18.36 HAL_LPTIM_IRQHandler

3.18.36.1 功能介绍

处理 LPTIM 中断请求。

3.18.36.2 接口定义

函数接口	void HAL_LPTIM_IRQHandler (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.37 HAL_LPTIM_CompareMatchCallback

3.18.37.1 功能介绍

在非阻塞模式下比较匹配回调。。

3.18.37.2 接口定义

函数接口	void HAL_LPTIM_CompareMatchCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.38 HAL_LPTIM_AutoReloadMatchCallback

3.18.38.1 功能介绍

在非阻塞模式下自动重新加载匹配回调。

3.18.38.2 接口定义

函数接口	void HAL_LPTIM_AutoReloadMatchCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.39 HAL_LPTIM_TriggerCallback

3.18.39.1 功能介绍

在非阻塞模式下触发检测回调。

3.18.39.2 接口定义

函数接口	void HAL_LPTIM_TriggerCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	

说明	
----	--

3.18.40 HAL_LPTIM_CompareWriteCallback

3.18.40.1 功能介绍

在非阻塞模式下比较写回调。

3.18.40.2 接口定义

函数接口	void HAL_LPTIM_CompareWriteCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.41 HAL_LPTIM_AutoReloadWriteCallback

3.18.41.1 功能介绍

在非阻塞模式下自动重新加载写回调。

3.18.41.2 接口定义

函数接口	void HAL_LPTIM_AutoReloadWriteCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.42 HAL_LPTIM_DirectionUpCallback

3.18.42.1 功能介绍

在非阻塞模式下，方向计数器从向下改变为向上回调。

3.18.42.2 接口定义

函数接口	void HAL_LPTIM_DirectionUpCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.43 HAL_LPTIM_DirectionDownCallback

3.18.43.1 功能介绍

在非阻塞模式下，方向计数器从向上变为向下回调。

3.18.43.2 接口定义

函数接口	void HAL_LPTIM_DirectionDownCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.44 HAL_LPTIM_DualInput1IdleErrorCallback

3.18.44.1 功能介绍

在非交编码模式下，通道 1 缺失错误回调。

3.18.44.2 接口定义

函数接口	void HAL_LPTIM_DualInput1IdleErrorCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.45 HAL_LPTIM_DualInput2IdleErrorCallback

3.18.45.1 功能介绍

在非交编码模式下，通道 2 缺失错误回调。

3.18.45.2 接口定义

函数接口	void HAL_LPTIM_DualInput2IdleErrorCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.46 HAL_LPTIM_DualB2BErrorCallback

3.18.46.1 功能介绍

在非交编码模式下，双通道脉冲连续错误回调。

3.18.46.2 接口定义

函数接口	void HAL_LPTIM_DualB2BErrorCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无

返回值	无
资源使用	
说明	

3.18.47 HAL_LPTIM_DualWaveFormErrorCallback

3.18.47.1 功能介绍

非交波形错误回调。

3.18.47.2 接口定义

函数接口	void HAL_LPTIM_DualWaveFormErrorCallback (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	无
资源使用	
说明	

3.18.48 HAL_LPTIM_GetState

3.18.48.1 功能介绍

获取 LPTIM 状态。

3.18.48.2 接口定义

函数接口	HAL_LPTIM_StateTypeDef HAL_LPTIM_GetState (LPTIM_HandleTypeDef * hlptim)
输入	LPTIM_HandleTypeDef * hlptim
输出	无
返回值	HAL_LPTIM_StateTypeDef
资源使用	
说明	

3.19 OPAMP 模块

宏定义	数值	含义
HAL_OPAMP_StateTypeDef		
HAL_OPAMP_STATE_RESET	0x00	复位
HAL_OPAMP_STATE_READY	0x01	就绪
HAL_OPAMP_STATE_CALIBBUSY	0x02	自动校准模式
HAL_OPAMP_STATE_BUSY	0x04	超时
HAL_OPAMP_STATE_BUSYLOCKED	0x05	锁定，只有系统重置允许重新配置 opamp

属性	类型	字段名	含义
----	----	-----	----

OPAMP_TypeDef			
读写	uint32_t	CSR	控制和状态寄存器
OPAMP_InitTypeDef			
读写	uint32_t	Mode	模式
读写	uint32_t	NonInvertingInput	正相输入
读写	uint32_t	InvertingInput	反相输入
读写	uint32_t	PgaGain	输入增益
读写	uint32_t	OutputMode	输出模式
读写	uint32_t	TrimmingSrc	校正源
OPAMP_HandleTypeDef			
读写	OPAMP_TypeDef	Instance	寄存器地址
读写	OPAMP_InitTypeDef	Init	初始化参数
读写	HAL_StatusTypeDef	Lock	锁定
读写	HAL_LockTypeDef	State	状态
读写	HAL_OPAMP_StateTypeDef	State	OPAMP 状态
读写	指针函数	void(*MspInitCallback)(struct __OPAMP_HandleTypeDef *hopamp)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback)(struct __OPAMP_HandleTypeDef *hopamp)	MSP 非初始化回调

3.19.1 HAL_OPAMP_Init

3.19.1.1 功能介绍

初始化 OPAMP。

3.19.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMP_Init (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.19.2 HAL_OPAMP_DeInit

3.19.2.1 功能介绍

清除 OPAMP 初始化参数。

3.19.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMP_DeInit (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.19.3 HAL_OPAMP_MspInit

3.19.3.1 功能介绍

初始化 OPAMP MSP。

3.19.3.2 接口定义

函数接口	void HAL_OPAMP_MspInit (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	无
资源使用	
说明	

3.19.4 HAL_OPAMP_MspDeInit

3.19.4.1 功能介绍

消除 OPAMP MSP 初始化参数。

3.19.4.2 接口定义

函数接口	void HAL_OPAMP_MspInit (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	无
资源使用	
说明	

3.19.5 HAL_OPAMP_Start

3.19.5.1 功能介绍

开启 OPAMP。

3.19.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMP_Start (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.19.6 HAL_OPAMP_Stop

3.19.6.1 功能介绍

停止 OPAMP。

3.19.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMP_Stop (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.19.7 HAL_OPAMP_SelfCalibrate

3.19.7.1 功能介绍

运行 OPAMP 的自校准。

3.19.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMP_SelfCalibrate (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.19.8 HAL_OPAMP_Lock

3.19.8.1 功能介绍

锁定 OPAMP。

3.19.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMP_Lock (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.19.9 HAL_OPAMP_GetState

3.19.9.1 功能介绍

获取 OPAMP 状态。

3.19.9.2 接口定义

函数接口	HAL_OPAMP_StateTypeDef HAL_OPAMP_GetState (OPAMP_HandleTypeDef * hopamp)
输入	OPAMP_HandleTypeDef * hopamp
输出	无
返回值	HAL_OPAMP_StateTypeDef
资源使用	
说明	

3.19.10 HAL_OPAMPEx_SelfCalibrateAll

3.19.10.1 功能介绍

运行 4 个 OPAMP 的自校准。

3.19.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_OPAMPEx_SelfCalibrateAll (OPAMP_HandleTypeDef * hopamp1, OPAMP_HandleTypeDef * hopamp2, OPAMP_HandleTypeDef * hopamp3, OPAMP_HandleTypeDef * hopamp4)
输入	OPAMP_HandleTypeDef * hopamp1 OPAMP_HandleTypeDef * hopamp2 OPAMP_HandleTypeDef * hopamp3 OPAMP_HandleTypeDef * hopamp4
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.20 PLA 模块

宏定义	数值	含义
HAL_PLA_StateTypeDef		
HAL_PLA_STATE_RESET	0x00	复位
HAL_PLA_STATE_READY	0x01	就绪
HAL_PLA_STATE_BUSY	0x02	超时

属性	类型	字段名	含义
PLA_TypeDef			
读写	uint32_t	CR	控制寄存器

读写	uint32_t	SR	标志寄存器
读写	uint32_t	CFGR0	PLU0 配置寄存器
读写	uint32_t	CFGR1	PLU1 配置寄存器
读写	uint32_t	CFGR2	PLU2 配置寄存器
读写	uint32_t	CFGR3	PLU3 配置寄存器
PLA_InitTypeDef			
读写	uint32_t	PortIndex	端口索引
读写	uint32_t	MUXAInputSel	MUXA 输入选择
读写	uint32_t	MUXBInputSel	MUXB 输入选择
读写	uint32_t	FunctionSel	功能选择
读写	uint32_t	OutputSel	输出选择
读写	FunctionalState	OutputEnable	输出使能
读写	uint32_t	DFFClockInverse	DFF 时钟反向模式
读写	uint32_t	DFFClockSource	DFF 时钟源
读写	uint32_t	TriggerMode	触发模式
PLA_HandleTypeDef			
读写	PLA_TypeDef	Instance	寄存器地址
读写	PLA_InitTypeDef	Init	初始化参数
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_PLA_StateTypeDef	State	状态
读写	uint32_t	ErrorCode	错误代码
读写	指针函数	void(*RisingTriggerCallback)(struct __PLA_HandleTypeDef *hpla)	上升沿触发回调
读写	指针函数	void(*FallingTriggerCallback)(struct __PLA_HandleTypeDef *hpla)	下降沿触发回调
读写	指针函数	void(*MspInitCallback)(struct __PLA_HandleTypeDef *hpla)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback)(struct	MSP 非初始

		__PLA_HandleTypeDef *hpla)	化回调
--	--	----------------------------	-----

3.20.1 HAL_PLA_Init

3.20.1.1 功能介绍

初始化 PLA。

3.20.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_PLA_Init (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.20.2 HAL_PLA_DeInit

3.20.2.1 功能介绍

清除 PLA 初始化参数。

3.20.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_PLA_DeInit (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.20.3 HAL_PLA_MspInit

3.20.3.1 功能介绍

初始化 PLA MSP。

3.20.3.2 接口定义

函数接口	void HAL_PLA_MspInit (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	无
资源使用	
说明	

3.20.4 HAL_PLA_MspDeInit

3.20.4.1 功能介绍

清除 PLA MSP 初始化参数。

3.20.4.2 接口定义

函数接口	void HAL_PLA_MspDeInit (PLA_HandleTypeDef * hpla)
------	---

输入	PLA_HandleTypeDef * hpla
输出	无
返回值	无
资源使用	
说明	

3.20.5 HAL_PLA_Start

3.20.5.1 功能介绍

开启 PLA。

3.20.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_PLA_Start (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.20.6 HAL_PLA_Stop

3.20.6.1 功能介绍

停止 PLA。

3.20.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_PLA_Stop (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.20.7 HAL_PLA_IRQHandler

3.20.7.1 功能介绍

处理 PLA 中断请求。

3.20.7.2 接口定义

函数接口	void HAL_PLA_IRQHandler (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	无
资源使用	
说明	

3.20.8 HAL_PLA_GetOutputLevel

3.20.8.1 功能介绍

获取 PLA 输出电平。

3.20.8.2 接口定义

函数接口	uint32_t HAL_PLA_GetOutputLevel (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	{ PLA_OUTPUT_LEVEL_LOW, PLA_OUTPUT_LEVEL_HIGH }
资源使用	
说明	

3.20.9 HAL_PLA_RisingTriggerCallback

3.20.9.1 功能介绍

PLA 上升沿触发回调。

3.20.9.2 接口定义

函数接口	void HAL_PLA_RisingTriggerCallback (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	无
资源使用	
说明	

3.20.10 HAL_PLA_FallingTriggerCallback

3.20.10.1 功能介绍

PLA 下降沿触发回调。

3.20.10.2 接口定义

函数接口	void HAL_PLA_FallingTriggerCallback (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	无
资源使用	
说明	

3.20.11 HAL_PLA_GetState

3.20.11.1 功能介绍

获取 PLA 状态。

3.20.11.2 接口定义

函数接口	HAL_PLA_StateTypeDef HAL_PLA_GetState (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla

输出	无
返回值	HAL_PLA_StateTypeDef
资源使用	
说明	

3.20.12 HAL_PLA_GetError

3.20.12.1 功能介绍

获取 PLA 错误。

3.20.12.2 接口定义

函数接口	uint32_t HAL_PLA_GetError (PLA_HandleTypeDef * hpla)
输入	PLA_HandleTypeDef * hpla
输出	无
返回值	错误值
资源使用	
说明	

3.21 RCC 模块

属性	类型	字段名	含义
RCC_PLLInitTypeDef			
读写	uint32_t	PLLState	PLL 状态
读写	uint32_t	PLLSource	PLL 时钟源
读写	uint32_t	PLLN	PLL 输入时钟的分频因子 N
读写	uint32_t	PLLM	PLL 输入时钟的倍频因子 M
读写	uint32_t	PLLQ	PLL VCO 的 Q 分频系数
读写	uint32_t	PLLR	PLL VCO 的 R 分频系数
RCC_OscInitTypeDef			
读写	uint32_t	OscillatorType	振荡器类型
读写	uint32_t	HSEState	HSE 状态
读写	uint32_t	HSEDrv	HSE 的驱动能力
读写	uint32_t	HSEInPin	HSE 的输入

			pin
读写	uint32_t	HSEOutPin	HSE 的输出 pin
读写	FunctionalState	HSEDiv	HSE 的分频 因子
读写	uint32_t	LSEStableCycle	LSE 的稳定 周期
读写	uint32_t	LSEDrv	LSE 的驱动 能力
读写	uint32_t	HSIState	触发模式
读写	uint32_t	HSIDiv	HSI 的分频 因子
读写	uint32_t	HSIStableCycle	HSI 的稳定 周期
读写	uint32_t	HSICalibrationValue	HIS 的校准 微调值
读写	uint32_t	LSIState	LSI 状态
读写	uint32_t	LSICalibrationValue	LSI 的校准 微调值
读写	uint32_t	MSIState	MSI 的状态
读写	uint32_t	MSIStableCycle	MSI 的稳定 周期
读写	uint32_t	MSICalibrationValue	MSI 的校准 微调值
读写	RCC_PLLInitTypeDef	PLL	PLL 参数
RCC_ClkInitTypeDef			
读写	uint32_t	ClockType	时钟类型
读写	uint32_t	SYSClkSource	系统时钟源
读写	uint32_t	AHBCLKDivider	AHB 时钟驱 动
读写	uint32_t	APB1CLKDivider	APB1 时钟驱 动
读写	uint32_t	APB2CLKDivider	APB2 时钟驱 动

3.21.1 HAL_RCC_DeInit

3.21.1.1 功能介绍

清除 RCC 初始化参数。

3.21.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RCC_DeInit (void)
输入	无
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.21.2 HAL_RCC_OscConfig

3.21.2.1 功能介绍

初始化 RCC 振荡器配置。

3.21.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RCC_OscConfig (RCC_OscInitTypeDef * RCC_OscInitStruct)
输入	RCC_OscInitTypeDef * RCC_OscInitStruct
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.21.3 HAL_RCC_ClockConfig

3.21.3.1 功能介绍

初始化 CPU、AHB 和 APB 总线时钟。

3.21.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RCC_ClockConfig (RCC_ClkInitTypeDef * RCC_ClkInitStruct, uint32_t FLatency)
输入	RCC_ClkInitTypeDef * RCC_ClkInitStruct uint32_t FLatency: { FLASH_LATENCY_0, FLASH_LATENCY_1, FLASH_LATENCY_2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.21.4 HAL_RCC_MCOConfig

3.21.4.1 功能介绍

选择输出在 MCO1 引脚上的时钟源。

3.21.4.2 接口定义

函数接口	void HAL_RCC_MCOConfig (uint32_t RCC_MCOx, uint32_t RCC_MCOSource, uint32_t RCC_MCODiv)
输入	uint32_t RCC_MCOx: {RCC_MCO1} uint32_t RCC_MCOSource: { RCC_MCO1SOURCE_NOCLOCK, RCC_MCO1SOURCE_SYSCLK, RCC_MCO1SOURCE_MSI, RCC_MCO1SOURCE_HSI, RCC_MCO1SOURCE_HSE, RCC_MCO1SOURCE_PLL, RCC_MCO1SOURCE_LSI, RCC_MCO1SOURCE_LSE } uint32_t RCC_MCODiv: { RCC_MCODIV_1, RCC_MCODIV_2, RCC_MCODIV_4, RCC_MCODIV_8, RCC_MCODIV_16, RCC_MCODIV_32, RCC_MCODIV_64, RCC_MCODIV_128 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.21.5 HAL_RCC_EnableHSECSS

3.21.5.1 功能介绍

使能 HSE 时钟安全系统。

3.21.5.2 接口定义

函数接口	void HAL_RCC_EnableHSECSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.6 HAL_RCC_EnablePLLCSS

3.21.6.1 功能介绍

使能 PLL 时钟安全系统。

3.21.6.2 接口定义

函数接口	void HAL_RCC_EnablePLLCSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.7 HAL_RCC_EnableLSECSS

3.21.7.1 功能介绍

使能 LSE 时钟安全系统。

3.21.7.2 接口定义

函数接口	void HAL_RCC_EnableLSECSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.8 HAL_RCC_DisableLSECSS

3.21.8.1 功能介绍

禁止使能 LSE 时钟安全系统。

3.21.8.2 接口定义

函数接口	void HAL_RCC_DisableLSECSS (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.9 HAL_RCC_GetSysClockFreq

3.21.9.1 功能介绍

获取 SYSCLK 频率。

3.21.9.2 接口定义

函数接口	uint32_t HAL_RCC_GetSysClockFreq (void)
输入	无
输出	无
返回值	频率值
资源使用	
说明	

3.21.10 HAL_RCC_GetHCLKFreq

3.21.10.1 功能介绍

获取 HCLK 频率。

3.21.10.2 接口定义

函数接口	uint32_t HAL_RCC_GetHCLKFreq (void)
输入	无
输出	无

返回值	频率值
资源使用	
说明	

3.21.11 HAL_RCC_GetPCLK1Freq

3.21.11.1 功能介绍

获取 PCLK1 频率。

3.21.11.2 接口定义

函数接口	uint32_t HAL_RCC_GetPCLK1Freq (void)
输入	无
输出	无
返回值	频率值
资源使用	
说明	

3.21.12 HAL_RCC_GetPCLK2Freq

3.21.12.1 功能介绍

获取 PCLK2 频率。

3.21.12.2 接口定义

函数接口	uint32_t HAL_RCC_GetPCLK2Freq (void)
输入	无
输出	无
返回值	频率值
资源使用	
说明	

3.21.13 HAL_RCC_GetOscConfig

3.21.13.1 功能介绍

获取 RCC 振荡器配置。

3.21.13.2 接口定义

函数接口	void HAL_RCC_GetOscConfig (RCC_OscInitTypeDef * RCC_OscInitStruct)
输入	RCC_OscInitTypeDef * RCC_OscInitStruct
输出	无
返回值	无
资源使用	
说明	

3.21.14 HAL_RCC_GetClockConfig

3.21.14.1 功能介绍

获取 CPU、AHB 和 APB 总线时钟配置。

3.21.14.2 接口定义

函数接口	void HAL_RCC_GetClockConfig (RCC_ClkInitTypeDef * RCC_ClkInitStruct, uint32_t * pFLatency)
输入	RCC_ClkInitTypeDef * RCC_ClkInitStruct uint32_t * pFLatency
输出	无
返回值	无
资源使用	
说明	

3.21.15 HAL_RCC_NMI_IRQHandler

3.21.15.1 功能介绍

处理 RCC 时钟安全系统中断请求。

3.21.15.2 接口定义

函数接口	void HAL_RCC_NMI_IRQHandler (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.16 HAL_RCC_HSECSSCallback

3.21.16.1 功能介绍

RCC HSE 时钟安全系统回调。

3.21.16.2 接口定义

函数接口	void HAL_RCC_HSECSSCallback (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.17 HAL_RCC_LSECSSCallback

3.21.17.1 功能介绍

RCC LSE 时钟安全系统回调。

3.21.17.2 接口定义

函数接口	void HAL_RCC_LSECSSCallback (void)
输入	无
输出	无

返回值	无
资源使用	
说明	

3.21.18 HAL_RCC_PLLCSSCallback

3.21.18.1 功能介绍

RCC PLL 时钟安全系统回调。

3.21.18.2 接口定义

函数接口	void HAL_RCC_PLLCSSCallback (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.19 HAL_RCCEX_PeriphCLKConfig

3.21.19.1 功能介绍

初始化 RCC 扩展外设时钟。

3.21.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RCCEX_PeriphCLKConfig (RCC_PeriphCLKInitTypeDef * PeriphClkInit)
输入	RCC_PeriphCLKInitTypeDef * PeriphClkInit: { RCC_PERIPHCLK_RTC, RCC_PERIPHCLK_ADC1, RCC_PERIPHCLK_TIM1, RCC_PERIPHCLK_I2C1, RCC_PERIPHCLK_USART1, RCC_PERIPHCLK_LPTIM1, RCC_PERIPHCLK_LPUART1 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.21.20 HAL_RCCEX_GetPeriphCLKConfig

3.21.20.1 功能介绍

根据内部 RCC 配置寄存器获取 RCC_ClkInitStruct。

3.21.20.2 接口定义

函数接口	void HAL_RCCEX_GetPeriphCLKConfig (RCC_PeriphCLKInitTypeDef * PeriphClkInit)
输入	RCC_PeriphCLKInitTypeDef * PeriphClkInit: { RCC_PERIPHCLK_RTC, RCC_PERIPHCLK_ADC1, RCC_PERIPHCLK_TIM1, RCC_PERIPHCLK_I2C1,

	RCC_PERIPHCLK_USART1, RCC_PERIPHCLK_LPTIM1, RCC_PERIPHCLK_LPUART1 }
输出	无
返回值	无
资源使用	
说明	

3.21.21 HAL_RCCEX_GetPeriphCLKFreq

3.21.21.1 功能介绍

从 PLL 返回带有时钟源的外设的时钟频率。

3.21.21.2 接口定义

函数接口	void HAL_RCCEX_GetPeriphCLKConfig (RCC_PeriphCLKInitTypeDef * PeriphClkInit)
输入	RCC_PeriphCLKInitTypeDef * PeriphClkInit: { RCC_PERIPHCLK_RTC, RCC_PERIPHCLK_ADC1, RCC_PERIPHCLK_TIM1, RCC_PERIPHCLK_I2C1, RCC_PERIPHCLK_USART1, RCC_PERIPHCLK_LPTIM1, RCC_PERIPHCLK_LPUART1 }
输出	无
返回值	时钟频率
资源使用	
说明	

3.21.22 HAL_RCCEX_EnableLSCO

3.21.22.1 功能介绍

使能低速时钟源输出在 LSCO 引脚上。

3.21.22.2 接口定义

函数接口	void HAL_RCCEX_EnableLSCO (uint32_t LSCOSource)
输入	uint32_t LSCOSource: { RCC_LSCOSOURCE_LSI, RCC_LSCOSOURCE_LSE }
输出	无
返回值	无
资源使用	
说明	

3.21.23 HAL_RCCEX_DisableLSCO

3.21.23.1 功能介绍

禁止使能低速时钟源输出在 LSCO 引脚上。

3.21.23.2 接口定义

函数接口	void HAL_RCCEX_DisableLSCO (uint32_t LSCOSource)
输入	uint32_t LSCOSource: { RCC_LSCOSOURCE_LSI, RCC_LSCOSOURCE_LSE }

输出	无
返回值	无
资源使用	
说明	

3.21.24 HAL_RCCEX_EnableRTCLowpower

3.21.24.1 功能介绍

使能 RTC 低功耗模式。

3.21.24.2 接口定义

函数接口	void HAL_RCCEX_EnableRTCLowpower (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.21.25 HAL_RCCEX_DisableRTCLowpower

3.21.25.1 功能介绍

禁止使能 RTC 低功耗模式。

3.21.25.2 接口定义

函数接口	void HAL_RCCEX_DisableRTCLowpower (void)
输入	无
输出	无
返回值	无
资源使用	
说明	

3.22 SPI 模块

宏定义	数值	含义
HAL_SPI_StateTypeDef		
HAL_SPI_STATE_RESET	0x00	复位
HAL_SPI_STATE_READY	0x01	就绪
HAL_SPI_STATE_BUSY	0x02	忙
HAL_SPI_STATE_BUSY_TX	0x03	发送忙
HAL_SPI_STATE_BUSY_RX	0x04	接收忙
HAL_SPI_STATE_BUSY_TX_RX	0x05	发送和接收忙
HAL_SPI_STATE_ERROR	0x06	错误
HAL_SPI_STATE_ABORT	0x07	中止

属性	类型	字段名	含义
SPI_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	FIFOC	FIFO 清空寄存器
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断标志清零寄存器
读写	uint32_t	DR	数据寄存器
SPI_InitTypeDef			
读写	uint32_t	Mode	模式
读写	uint32_t	ClockPolarity	时钟极性
读写	uint32_t	ClockPhase	时钟相位
读写	uint32_t	NSS	NSS 模式
读写	uint32_t	BaudRatePrescaler	波特率分频
读写	uint32_t	FirstBit	第一个传输位
读写	uint32_t	NSSO	NSS 输出
读写	uint32_t	TxFIFODepth	发送 FIFO 阈值
读写	uint32_t	RxFIFODepth	接收 FIFO 阈值
SPI_HandleTypeDef			
读写	SPI_TypeDef	Instance	寄存器地址
读写	SPI_InitTypeDef	Init	初始化参数
读写	uint8_t	pTxBuffPtr	TX 传输缓冲区的指针
读写	uint16_t	TxXferSize	传输发送大小
读写	uint16_t	TxXferCount	传输发送计数器

读写	uint16_t	pRxBuffPtr	RX 传输缓冲区的指针
读写	uint16_t	RxXferSize	传输接收大小
读写	uint16_t	RxXferCount	传输接收计数器
读写	uint16_t	NbRxDataToProcess	在 RX ISR 执行期间要处理的数据数
读写	uint16_t	NbTxDataToProcess	在 TX ISR 执行期间要处理的数据数
读写	指针函数	void (*RxISR)(struct __SPI_HandleTypeDef *hspi)	Rx ISR 函数指针
读写	指针函数	void (*TxISR)(struct __SPI_HandleTypeDef *hspi)	Tx ISR 函数指针
读写	DMA_HandleTypeDef	hdmatx	DMA 句柄
读写	DMA_HandleTypeDef	hdmarx	DMA 句柄
读写	HAL_LockTypeDef	Lock	锁定
读写	HAL_SPI_StateTypeDef	State	SPI 状态
读写	uint32_t	ErrorCode	错误代码
读写	指针函数	void(*TxCpltCallback)(struct __SPI_HandleTypeDef *hspi)	发送传输完成回调
读写	指针函数	void(*RxCpltCallback)(struct __SPI_HandleTypeDef *hspi)	接收传输完成回调
读写	指针函数	void(*TxRxCpltCallback)(struct __SPI_HandleTypeDef *hspi)	发送接收完成回调
读写	指针函数	void(*TxHalfCpltCallback)(struct __SPI_HandleTypeDef *hspi)	发送传输完成一半回调
读写	指针函数	void(*RxHalfCpltCallback)(struct __SPI_HandleTypeDef *hspi)	接收传输完成一半回调
读写	指针函数	void(*TxRxHalfCpltCallback)(struct __SPI_HandleTypeDef *hspi)	发送接收完成一半回调
读写	指针函数	void(*ErrorCallback)(struct __SPI_HandleTypeDef *hspi)	错误回调

读写	指针函数	void(*AbortCpltCallback)(struct __SPI_HandleTypeDef *hspi)	中止完成回调
读写	指针函数	void(*RxFifoFullCallback)(struct __SPI_HandleTypeDef *hspi)	接收 FIFO 满回调
读写	指针函数	void(*TxFifoEmptyCallback)(struct __SPI_HandleTypeDef *hspi)	发送 FIFO 空回调
读写	指针函数	void(*MspInitCallback)(struct __SPI_HandleTypeDef *hspi)	MSP 初始化回调
读写	指针函数	void(*MspDeInitCallback)(struct __SPI_HandleTypeDef *hspi)	MSP 非初始化回调

3.22.1 HAL_SPI_Init

3.22.1.1 功能介绍

初始化 SPI。

3.22.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Init (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.2 HAL_SPI_DeInit

3.22.2.1 功能介绍

清除 SPI 初始化参数。

3.22.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_DeInit (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.3 HAL_SPI_MspInit

3.22.3.1 功能介绍

初始化 SPI MSP。

3.22.3.2 接口定义

函数接口	void HAL_SPI_MspInit (SPI_HandleTypeDef * hspi)
------	---

输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.4 HAL_SPI_MspDeInit

3.22.4.1 功能介绍

清除 SPI MSP 初始化参数。

3.22.4.2 接口定义

函数接口	void HAL_SPI_MspDeInit (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.5 HAL_SPI_Transmit

3.22.5.1 功能介绍

SPI 以阻塞方式发送数据。

3.22.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Transmit (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	SPI_HandleTypeDef * hspi uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.6 HAL_SPI_Receive

3.22.6.1 功能介绍

SPI 以阻塞方式接收数据。

3.22.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Receive (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	SPI_HandleTypeDef * hspi uint8_t * pData uint16_t Size uint32_t Timeout

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.7 HAL_SPI_TransmitReceive

3.22.7.1 功能介绍

SPI 以阻塞方式发送和接收数据。

3.22.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_TransmitReceive (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)
输入	SPI_HandleTypeDef * hspi uint8_t * pTxData uint8_t * pRxData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.8 HAL_SPI_Transmit_IT

3.22.8.1 功能介绍

SPI 使用中断以非阻塞模式发送数据。

3.22.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Transmit_IT (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
输入	SPI_HandleTypeDef * hspi uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.9 HAL_SPI_Receive_IT

3.22.9.1 功能介绍

SPI 使用中断以非阻塞模式接收数据。

3.22.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Receive_IT (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
------	---

输入	SPI_HandleTypeDef * hspi uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.10 HAL_SPI_TransmitReceive_IT

3.22.10.1 功能介绍

SPI 使用中断以非阻塞模式发送和接收数据。

3.22.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_TransmitReceive_IT (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
输入	SPI_HandleTypeDef * hspi uint8_t * pTxData uint8_t * pRxData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.11 HAL_SPI_Transmit_DMA

3.22.11.1 功能介绍

SPI 使用 DMA 以非阻塞模式发送数据。

3.22.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Transmit_DMA (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
输入	SPI_HandleTypeDef * hspi uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.12 HAL_SPI_Receive_DMA

3.22.12.1 功能介绍

SPI 使用 DMA 以非阻塞模式接收数据。

3.22.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Receive_DMA (SPI_HandleTypeDef * hspi, uint8_t * pData, uint16_t Size)
输入	SPI_HandleTypeDef * hspi uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.13 HAL_SPI_TransmitReceive_DMA

3.22.13.1 功能介绍

SPI 使用 DMA 以非阻塞模式发送和接收数据。

3.22.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_TransmitReceive_DMA (SPI_HandleTypeDef * hspi, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
输入	SPI_HandleTypeDef * hspi uint8_t * pTxData uint8_t * pRxData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.14 HAL_SPI_DMAMPause

3.22.14.1 功能介绍

暂停 DMA 传输。

3.22.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_DMAMPause (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.15 HAL_SPI_DMAResume

3.22.15.1 功能介绍

恢复 DMA 传输。

3.22.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_DMAResume (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.16 HAL_SPI_DMAStop

3.22.16.1 功能介绍

停止 DMA 传输。

3.22.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_DMAStop (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.17 HAL_SPI_Abort

3.22.17.1 功能介绍

中止正在进行的传输(阻塞模式)。

3.22.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Abort (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.18 HAL_SPI_Abort_IT

3.22.18.1 功能介绍

中止正在进行的传输(中断模式)。

3.22.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_SPI_Abort_IT (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.19 HAL_SPI_IRQHandler

3.22.19.1 功能介绍

处理 SPI 中断请求。

3.22.19.2 接口定义

函数接口	void HAL_SPI_IRQHandler (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.22.20 HAL_SPI_TxCpltCallback

3.22.20.1 功能介绍

SPI 发送完成回调。

3.22.20.2 接口定义

函数接口	void HAL_SPI_TxCpltCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.21 HAL_SPI_RxCpltCallback

3.22.21.1 功能介绍

SPI 接收完成回调。

3.22.21.2 接口定义

函数接口	void HAL_SPI_RxCpltCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.22 HAL_SPI_TxRxCpltCallback

3.22.22.1 功能介绍

SPI 发送和接收完成回调。

3.22.22.2 接口定义

函数接口	void HAL_SPI_TxRxCpltCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无

返回值	无
资源使用	
说明	

3.22.23 HAL_SPI_TxHalfCpltCallback

3.22.23.1 功能介绍

SPI 发送完成一半回调。

3.22.23.2 接口定义

函数接口	void HAL_SPI_TxCpltHalfCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.24 HAL_SPI_RxHalfCpltCallback

3.22.24.1 功能介绍

SPI 接收完成一半回调。

3.22.24.2 接口定义

函数接口	void HAL_SPI_RxHalfCpltCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.25 HAL_SPI_TxRxHalfCpltCallback

3.22.25.1 功能介绍

SPI 发送和接收完成一半回调。

3.22.25.2 接口定义

函数接口	void HAL_SPI_TxRxHalfCpltCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.26 HAL_SPI_ErrorCallback

3.22.26.1 功能介绍

SPI 错误回调。

3.22.26.2 接口定义

函数接口	void HAL_SPI_ErrorCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.27 HAL_SPI_AbortCpltCallback

3.22.27.1 功能介绍

SPI 中止完成回调。

3.22.27.2 接口定义

函数接口	void HAL_SPI_AbortCpltCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.28 HAL_SPI_RxFifoFullCallback

3.22.28.1 功能介绍

SPI 接收 FIFO 满回调。

3.22.28.2 接口定义

函数接口	void HAL_SPI_RxFifoFullCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.29 HAL_SPI_TxFifoEmptyCallback

3.22.29.1 功能介绍

SPI 发送 FIFO 空回调。

3.22.29.2 接口定义

函数接口	void HAL_SPI_TxFifoEmptyCallback (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	无
资源使用	
说明	

3.22.30 HAL_SPI_GetState

3.22.30.1 功能介绍

获取 SPI 状态。

3.22.30.2 接口定义

函数接口	HAL_SPI_StateTypeDef HAL_SPI_GetState (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	HAL_SPI_StateTypeDef
资源使用	
说明	

3.22.31 HAL_SPI_GetError

3.22.31.1 功能介绍

获取 SPI 错误。

3.22.31.2 接口定义

函数接口	uint32_t HAL_SPI_GetError (SPI_HandleTypeDef * hspi)
输入	SPI_HandleTypeDef * hspi
输出	无
返回值	错误值
资源使用	
说明	

3.23 TIM 模块

宏定义	数值	含义
HAL_TIM_StateTypeDef		
HAL_TIM_STATE_RESET	0x00	复位
HAL_TIM_STATE_READY	0x01	就绪
HAL_TIM_STATE_BUSY	0x02	忙
HAL_TIM_STATE_TIMEOUT	0x03	超时
HAL_TIM_STATE_ERROR	0x04	错误
HAL_TIM_ChannelStateTypeDef		
HAL_TIM_CHANNEL_STATE_RESET	0x00	TIM 通道复位
HAL_TIM_CHANNEL_STATE_READY	0x01	TIM 通道就绪
HAL_TIM_CHANNEL_STATE_BUSY	0x02	TIM 通道忙
HAL_TIM_DMABurstStateTypeDef		
HAL_DMA_BURST_STATE_RESET	0x00	DMA burst 复位
HAL_DMA_BURST_STATE_READY	0x01	DMA burst 就绪

HAL_DMA_BURST_STATE_BUSY	0x02	DMA burst 忙
HAL_TIM_ActiveChannel		
HAL_TIM_ACTIVE_CHANNEL_1	0x00	激活通道 1
HAL_TIM_ACTIVE_CHANNEL_2	0x02	激活通道 2
HAL_TIM_ACTIVE_CHANNEL_3	0x04	激活通道 3
HAL_TIM_ACTIVE_CHANNEL_4	0x06	激活通道 4
HAL_TIM_ACTIVE_CHANNEL_5	0x07	激活通道 5
HAL_TIM_ACTIVE_CHANNEL_6	0x08	激活通道 6
HAL_TIM_ACTIVE_CHANNEL_CLEARED	0x10	清除所有激活通道

属性	类型	字段名	含义
TIM_TypeDef			
读 写	uint32_t	CR1	控 制 寄 存 器 1
读 写	uint32_t	CR2	控 制 寄 存 器 2
读 写	uint32_t	SMCR	从 模 式 控 制 寄 存 器
读 写	uint32_t	DIER	DMA/ 中 断 使 能 寄 存 器
读 写	uint32_t	SR	状 态 寄 存 器
读 写	uint32_t	EGR	事 件 产 生 寄 存 器

读 写	uint32_t	CCMR1_Output	捕获/ 比较 模式 寄存 器 1 输出 比较 模式
读 写	uint32_t	CCMR1_Input	捕获/ 比较 模式 寄存 器 1 输入 捕获 模式
读 写	uint32_t	CCMR2_Output	捕获/ 比较 模式 寄存 器 2 输出 比较 模式
读 写	uint32_t	CCMR2_Input	捕获/ 比较 模式 寄存 器 2 输入 捕获 模式
读 写	uint32_t	CCER	捕获/ 比较 使 能

			寄存器
读写	uint32_t	CNT	计数值寄存器
读写	uint32_t	PSC	预分频寄存器
读写	uint32_t	ARR	自动重载值寄存器
读写	uint32_t	RCR	重复计数器寄存器
读写	uint32_t	CCR1	捕获/比较寄存器 1
读写	uint32_t	CCR2	捕获/比较寄存器 2
读写	uint32_t	CCR3	捕获/比较寄存器 3
读写	uint32_t	CCR4	捕获/比较寄存器 4
读写	uint32_t	BDTR	断路和死

			区 寄 存 器
读 写	uint32_t	DCR	DMA 控 制 寄 存 器
读 写	uint32_t	DMAR	DMA 全 传 输 地 址 寄 存 器
读 写	uint32_t	OR1	配 置 寄 存 器
读 写	uint32_t	CCMR3_Output	捕 获 / 比 较 模 式 寄 存 器 3
读 写	uint32_t	CCR5	捕 获 / 比 较 寄 存 器 5
读 写	uint32_t	CCR6	捕 获 / 比 较 寄 存 器 6
读 写	uint32_t	AF1	轮 换 功 能 寄 存 器 1
读 写	uint32_t	AF2	轮 换 功 能 寄 存 器 3

读 写	uint32_t	TISEL	定 时 器 输 入 选 择 寄 存 器
TIM_Base_InitTypeDef			
读 写	uint32_t	Prescaler	预 分 频 器 值
读 写	uint32_t	CounterMode	计 数 模 式
读 写	uint32_t	Period	周 期
读 写	uint32_t	ClockDivision	时 钟 分 频
读 写	uint32_t	RepetitionCounter	重 复 计 数 器 值
读 写	uint32_t	AutoReloadPreload	自 动 预 装 载
TIM_OC_InitTypeDef			
读 写	uint32_t	OCMode	输 出 比 较 模 式
读 写	uint32_t	Pulse	脉 冲 值。
读 写	uint32_t	OCPolarity	输 出 极 性
读 写	uint32_t	OCNPolarity	互 补 输 出 极 性
读 写	uint32_t	OCFastMode	快 速 模 式
读	uint32_t	OCIdleState	输 出

写			空闲状态
读写	uint32_t	OCNIdleState	互补输出空闲状态
TIM_OnePulse_InitTypeDef			
读写	uint32_t	OCMode	输出比较模式
读写	uint32_t	Pulse	脉冲值。
读写	uint32_t	OCPolarity	输出极性
读写	uint32_t	OCNPolarity	互补输出极性
读写	uint32_t	OCIdleState	输出空闲状态
读写	uint32_t	OCNIdleState	互补输出空闲状态
读写	uint32_t	ICPolarity	输入极性
读写	uint32_t	ICSelection	输入选择
读写	uint32_t	ICFilter	输入捕获滤波器
TIM_IC_InitTypeDef			
读写	uint32_t	ICPolarity	输入极性

读 写	uint32_t	ICSelection	输入 选择
读 写	uint32_t	ICPrescaler	输入 捕获 预分 频器
读 写	uint32_t	ICFilter	输入 捕获 滤波 器
TIM_ENCODER_InitTypeDef			
读 写	uint32_t	EncoderMode	从模 式选 择
读 写	uint32_t	IC1Polarity	输入1 极性
读 写	uint32_t	ICSelection	输入1 选择
读 写	uint32_t	IC1Prescaler	输入1 捕获 预分 频器
读 写	uint32_t	IC1Filter	输入1 捕获 滤波 器
读 写	uint32_t	IC2Polarity	输入2 极性
读 写	uint32_t	ICSelection	输入2 选择
读 写	uint32_t	IC2Prescaler	输入2 捕获 预分 频器
读	uint32_t	IC2Filter	输入2

写			捕获滤波器
TIM_ClockConfigTypeDef			
读写	uint32_t	ClockSource	时钟源
读写	uint32_t	ClockPolarity	时钟极性
读写	uint32_t	ClockPrescaler	时钟分频
读写	uint32_t	ClockFilter	时钟滤波
TIM_ClearInputConfigTypeDef			
读写	uint32_t	ClearInputState	清除输入状态
读写	uint32_t	ClearInputSource	清除输入源
读写	uint32_t	ClearInputPolarity	清除输入极性
读写	uint32_t	ClearInputPrescaler	清除输入分频值
读写	uint32_t	ClearInputFilter	清除输入滤波
TIM_MasterConfigTypeDef			
读写	uint32_t	MasterOutputTrigger	主模式输出触发源
读	uint32_t	MasterOutputTrigger2	主模

写			式 输出 触发源 2
读写	uint32_t	MasterSlaveMode	主 从 模式
TIM_SlaveConfigTypeDef			
读写	uint32_t	SlaveMode	从 模 式
读写	uint32_t	InputTrigger	输 入 触 发 源
读写	uint32_t	TriggerPolarity	触 发 源 极 性
读写	uint32_t	TriggerPrescaler	触 发 源 分 频 值
读写	uint32_t	TriggerFilter	触 发 源 滤 波
TIM_BreakDeadTimeConfigTypeDef			
读写	uint32_t	OffStateRunMode	运 行 模 式 下 的 关 闭 状 态 选 择
读写	uint32_t	OffStateIDLEMode	空 闲 模 式 下 的 关 闭 状 态 选 择
读写	uint32_t	LockLevel	锁 定 级 别

读 写	uint32_t	DeadTime	死区 发生 时间
读 写	uint32_t	BreakState	断 路 状 态
读 写	uint32_t	BreakPolarity	断 路 极 性
读 写	uint32_t	BreakFilter	断 路 滤 波
读 写	uint32_t	BreakAFMode	断 路 复 用 模 式
读 写	uint32_t	Break2State	断 路 2 状 态
读 写	uint32_t	Break2Polarity	断 路 2 极 性
读 写	uint32_t	Break2Filter	断 路 2 滤 波
读 写	uint32_t	Break2AFMode	断 路 2 复 用 模 式
读 写	uint32_t	AutomaticOutput	自 动 输 出
TIM_HandleTypeDef			
读 写	TIM_TypeDef	Instance	寄 存 器 地 址
读 写	TIM_Base_InitTypeDef	Init	初 始 化 基 础 参 数
读 写	HAL_TIM_ActiveChannel	Channel	激 活 通 道
读 写	DMA_HandleTypeDef	hdma	DMA 句柄

读 写	HAL_LockTypeDef	Lock	锁定
读 写	HAL_TIM_StateTypeDef	State	TIM 状态
读 写	HAL_TIM_ChannelStateTypeDef	ChannelState	通道 操作 状态
读 写	HAL_TIM_ChannelStateTypeDef	ChannelNState	所有 通道 操作 状态
读 写	HAL_TIM_DMABurstStateTypeDef	DMABurstState	DMA Burst 状态
读 写	指针函数	void(*Base_MspInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM 基础 MSP 初始化 回调
读 写	指针函数	void(*Base_MspDeInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM 基础 MSP 非初始化 回调
读 写	指针函数	void(*IC_MspInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM IC MSP 初始化 回调
读 写	指针函数	void(*IC_MspDeInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM IC MSP 非

			初始化回调
读写	指针函数	<code>void(*OC_MspInitCallback)(struct __TIM_HandleTypeDef *htim)</code>	TIM OC MSP 初始化回调
读写	指针函数	<code>void(*OC_MspDeInitCallback)(struct __TIM_HandleTypeDef *htim)</code>	TIM OC MSP 非初始化回调
读写	指针函数	<code>void(*PWM_MspInitCallback)(struct __TIM_HandleTypeDef *htim)</code>	TIM PWM MSP 初始化回调
读写	指针函数	<code>void(*PWM_MspDeInitCallback)(struct __TIM_HandleTypeDef *htim)</code>	TIM PWM MSP 非初始化回调
读写	指针函数	<code>void(*OnePulse_MspInitCallback)(struct __TIM_HandleTypeDef *htim)</code>	TIM 单脉冲 MSP 初始化回调
读	指针函数	<code>void(*OnePulse_MspDeInitCallback)(struct</code>	TIM

写		__TIM_HandleTypeDef *htim)	单脉冲 MSP 非初始化 回调
读写	指针函数	void(*Encoder_MspInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM 编码器 MSP 初始化 回调
读写	指针函数	void(*Encoder_MspDeInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM 编码器 MSP 非初始化 回调
读写	指针函数	void(*HallSensor_MspInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM 霍尔 传感器 接口 MSP 初始化 回调
读写	指针函数	void(*HallSensor_MspDeInitCallback)(struct __TIM_HandleTypeDef *htim)	TIM 霍尔 传感器 接口 MSP 非初

			始 化 回 调
读 写	指针函数	void(*PeriodElapsedCallback)(struct __TIM_HandleTypeDef *htim)	周 期 经 过 回 调
读 写	指针函数	void(* PeriodElapsedHalfCpltCallback)(struct __TIM_HandleTypeDef *htim)	周 期 经 过 一 半 回 调
读 写	指针函数	void(*TriggerCallback)(struct __TIM_HandleTypeDef *htim)	触 发 回 调
读 写	指针函数	void(*TriggerHalfCpltCallback)(struct __TIM_HandleTypeDef *htim)	触 发 完 成 一 半 回 调
读 写	指针函数	void(*IC_CaptureCallback)(struct __TIM_HandleTypeDef *htim)	输 入 捕 获 回 调
读 写	指针函数	void(*IC_CaptureHalfCpltCallback)(struct __TIM_HandleTypeDef *htim)	输 入 捕 获 完 成 一 半 回 调
读 写	指针函数	void(*OC_DelayElapsedCallback)(struct __TIM_HandleTypeDef *htim)	输 出 比 较 延 迟 执 行 回 调
读 写	指针函数	void(*PWM_PulseFinishedCallback)(struct __TIM_HandleTypeDef *htim)	PWM 脉 冲 完 成 回 调
读 写	指针函数	void(*PWM_PulseFinishedHalfCpltCallback) (struct __TIM_HandleTypeDef *htim)	PWM 脉 冲 完 成

			一半回调
读写	指针函数	void(*ErrorCallback)(struct __TIM_HandleTypeDef *htim)	错误回调
读写	指针函数	void(*CommutationCallback)(struct __TIM_HandleTypeDef *htim)	通信回调
读写	指针函数	void(*CommutationHalfCpltCallback)(struct __TIM_HandleTypeDef *htim)	通信完成一半回调
读写	指针函数	void(*BreakCallback)(struct __TIM_HandleTypeDef *htim)	断路回调
读写	指针函数	void(*Break2Callback)(struct __TIM_HandleTypeDef *htim)	断路2回调

3.23.1 HAL_TIM_OC_ConfigChannel

3.23.1.1 功能介绍

初始化 TIM 输出比较通道。

3.23.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_ConfigChannel (TIM_HandleTypeDef * htim, TIM_OC_InitTypeDef * sConfig, uint32_t Channel)
输入	TIM_HandleTypeDef * htim TIM_OC_InitTypeDef * sConfig uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.2 HAL_TIM_PWM_ConfigChannel

3.23.2.1 功能介绍

初始化 TIM PWM 通道。

3.23.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_ConfigChannel (TIM_HandleTypeDef * htim, TIM_OC_InitTypeDef * sConfig, uint32_t Channel)
------	--

输入	TIM_HandleTypeDef * htim TIM_OC_InitTypeDef * sConfig uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.3 HAL_TIM_IC_ConfigChannel

3.23.3.1 功能介绍

初始化 TIM 输入比较通道。

3.23.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_ConfigChannel (TIM_HandleTypeDef * htim, TIM_IC_InitTypeDef * sConfig, uint32_t Channel)
输入	TIM_HandleTypeDef * htim TIM_IC_InitTypeDef * sConfig uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.4 HAL_TIM_OnePulse_ConfigChannel

3.23.4.1 功能介绍

初始化 TIM 单脉冲通道。

3.23.4.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_ConfigChannel (TIM_HandleTypeDef * htim, TIM_OnePulse_InitTypeDef * sConfig, uint32_t OutputChannel, uint32_t InputChannel)
输入	TIM_HandleTypeDef * htim TIM_OnePulse_InitTypeDef * sConfig uint32_t OutputChannel: { TIM_CHANNEL_1, TIM_CHANNEL_2 } uint32_t InputChannel: { TIM_CHANNEL_1, TIM_CHANNEL_2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.23.5 HAL_TIM_ConfigOCrefClear

3.23.5.1 功能介绍

配置 OCref 清除特性。

3.23.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_ConfigOCrefClear (TIM_HandleTypeDef * htim, TIM_ClearInputConfigTypeDef * sClearInputConfig, uint32_t Channel)
输入	TIM_HandleTypeDef * htim TIM_ClearInputConfigTypeDef * sClearInputConfig uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.6 HAL_TIM_ConfigClockSource

3.23.6.1 功能介绍

配置时钟源。

3.23.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_ConfigClockSource (TIM_HandleTypeDef * htim, TIM_ClockConfigTypeDef * sClockSourceConfig)
输入	TIM_HandleTypeDef * htim TIM_ClockConfigTypeDef * sClockSourceConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.7 HAL_TIM_ConfigTI1Input

3.23.7.1 功能介绍

选择连接到 TI1 输入端的信号。

3.23.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_ConfigTI1Input (TIM_HandleTypeDef * htim, uint32_t TI1_Selection)
输入	TIM_HandleTypeDef * htim uint32_t TI1_Selection: { TIM_TI1SELECTION_CH1, TIM_TI1SELECTION_XORCOMBINATION }

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.8 HAL_TIM_SlaveConfigSynchro

3.23.8.1 功能介绍

配置 TIM 为从模式。

3.23.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchro (TIM_HandleTypeDef * htim, TIM_SlaveConfigTypeDef * sSlaveConfig)
输入	TIM_HandleTypeDef * htim TIM_SlaveConfigTypeDef * sSlaveConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.9 HAL_TIM_SlaveConfigSynchro_IT

3.23.9.1 功能介绍

在中断模式下配置 TIM 为从模式。

3.23.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_SlaveConfigSynchro_IT (TIM_HandleTypeDef * htim, TIM_SlaveConfigTypeDef * sSlaveConfig)
输入	TIM_HandleTypeDef * htim TIM_SlaveConfigTypeDef * sSlaveConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.10 HAL_TIM_DMABurst_WriteStart

3.23.10.1 功能介绍

配置 DMA 突发以将数据从内存传输到 TIM 外设。

3.23.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStart (TIM_HandleTypeDef * htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t BurstLength)
输入	TIM_HandleTypeDef * htim uint32_t BurstBaseAddress: { TIM_DMABASE_CR1, TIM_DMABASE_CR2, TIM_DMABASE_SMCR, TIM_DMABASE_DIER, TIM_DMABASE_SR, TIM_DMABASE_EGR,

	<pre>TIM_DMABASE_CCMR1, TIM_DMABASE_CCMR2, TIM_DMABASE_CCER, TIM_DMABASE_CNT, TIM_DMABASE_PSC, TIM_DMABASE_ARR, TIM_DMABASE_RCR, TIM_DMABASE_CCR1, TIM_DMABASE_CCR2, TIM_DMABASE_CCR3 TIM_DMABASE_CCR4, TIM_DMABASE_BDTR, TIM_DMABASE_OR1, TIM_DMABASE_CCMR3, TIM_DMABASE_CCR5, TIM_DMABASE_CCR6, TIM_DMABASE_AF1, TIM_DMABASE_AF2, TIM_DMABASE_TISEL } uint32_t BurstRequestSrc: { TIM_DMA_UPDATE, TIM_DMA_CC1, TIM_DMA_CC2, TIM_DMA_CC3, TIM_DMA_CC4, TIM_DMA_COM, TIM_DMA_TRIGGER } uint32_t * BurstBuffer uint32_t BurstLength: [TIM_DMABURSTLENGTH_1TRANSFER, TIM_DMABURSTLENGTH_18TRANSFERS]</pre>
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.11 HAL_TIM_DMABurst_MultiWriteStart

3.23.11.1 功能介绍

配置 DMA Burst 将多个数据从内存传输到 TIM 外设。

3.23.11.2 接口定义

函数接口	<pre>HAL_StatusTypeDef HAL_TIM_DMABurst_MultiWriteStart (TIM_HandleTypeDef * htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t BurstLength, uint32_t DataLength)</pre>
输入	<pre>TIM_HandleTypeDef * htim uint32_t BurstBaseAddress: { TIM_DMABASE_CR1, TIM_DMABASE_CR2, TIM_DMABASE_SMCR, TIM_DMABASE_DIER, TIM_DMABASE_SR, TIM_DMABASE_EGR, TIM_DMABASE_CCMR1, TIM_DMABASE_CCMR2, TIM_DMABASE_CCER, TIM_DMABASE_CNT, TIM_DMABASE_PSC, TIM_DMABASE_ARR, TIM_DMABASE_RCR, TIM_DMABASE_CCR1, TIM_DMABASE_CCR2, TIM_DMABASE_CCR3 TIM_DMABASE_CCR4, TIM_DMABASE_BDTR, TIM_DMABASE_OR1, TIM_DMABASE_CCMR3, TIM_DMABASE_CCR5, TIM_DMABASE_CCR6,</pre>

	TIM_DMABASE_AF1, TIM_DMABASE_AF2, TIM_DMABASE_TISEL } uint32_t BurstRequestSrc: { TIM_DMA_UPDATE, TIM_DMA_CC1, TIM_DMA_CC2, TIM_DMA_CC3, TIM_DMA_CC4, TIM_DMA_COM, TIM_DMA_TRIGGER } uint32_t * BurstBuffer uint32_t BurstLength: [TIM_DMABURSTLENGTH_1TRANSFER, TIM_DMABURSTLENGTH_18TRANSFERS] uint32_t DataLength: [1, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.12 HAL_TIM_DMABurst_WriteStop

3.23.12.1 功能介绍

停止 TIM DMA Burst 模式。

3.23.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_DMABurst_WriteStop (TIM_HandleTypeDef * htim, uint32_t BurstRequestSrc)
输入	TIM_HandleTypeDef * htim uint32_t BurstRequestSrc: { TIM_DMA_UPDATE, TIM_DMA_CC1, TIM_DMA_CC2, TIM_DMA_CC3, TIM_DMA_CC4, TIM_DMA_COM, TIM_DMA_TRIGGER }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.13 HAL_TIM_DMABurst_ReadStart

3.23.13.1 功能介绍

配置 DMA Burst 将数据从 TIM 外设传输到内存。

3.23.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStart (TIM_HandleTypeDef * htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t BurstLength)
输入	TIM_HandleTypeDef * htim uint32_t BurstBaseAddress: { TIM_DMABASE_CR1, TIM_DMABASE_CR2, TIM_DMABASE_SMCR, TIM_DMABASE_DIER,

	<pre> TIM_DMABASE_SR, TIM_DMABASE_EGR, TIM_DMABASE_CCMR1, TIM_DMABASE_CCMR2, TIM_DMABASE_CCER, TIM_DMABASE_CNT, TIM_DMABASE_PSC, TIM_DMABASE_ARR, TIM_DMABASE_RCR, TIM_DMABASE_CCR1, TIM_DMABASE_CCR2, TIM_DMABASE_CCR3 TIM_DMABASE_CCR4, TIM_DMABASE_BDTR, TIM_DMABASE_OR1, TIM_DMABASE_CCMR3, TIM_DMABASE_CCR5, TIM_DMABASE_CCR6, TIM_DMABASE_AF1, TIM_DMABASE_AF2, TIM_DMABASE_TISEL } uint32_t BurstRequestSrc: { TIM_DMA_UPDATE, TIM_DMA_CC1, TIM_DMA_CC2, TIM_DMA_CC3, TIM_DMA_CC4, TIM_DMA_COM, TIM_DMA_TRIGGER } uint32_t * BurstBuffer uint32_t BurstLength: [TIM_DMABURSTLENGTH_1TRANSFER, TIM_DMABURSTLENGTH_18TRANSFERS] </pre>
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.14 HAL_TIM_DMABurst_MultiReadStart

3.23.14.1 功能介绍

配置 DMA Burst 将数据从 TIM 外设传输到内存。

3.23.14.2 接口定义

函数接口	<pre> HAL_StatusTypeDef HAL_TIM_DMABurst_MultiReadStart (TIM_HandleTypeDef * htim, uint32_t BurstBaseAddress, uint32_t BurstRequestSrc, uint32_t * BurstBuffer, uint32_t BurstLength, uint32_t DataLength) </pre>
输入	<pre> TIM_HandleTypeDef * htim uint32_t BurstBaseAddress: { TIM_DMABASE_CR1, TIM_DMABASE_CR2, TIM_DMABASE_SMCR, TIM_DMABASE_DIER, TIM_DMABASE_SR, TIM_DMABASE_EGR, TIM_DMABASE_CCMR1, TIM_DMABASE_CCMR2, TIM_DMABASE_CCER, TIM_DMABASE_CNT, TIM_DMABASE_PSC, TIM_DMABASE_ARR, TIM_DMABASE_RCR, TIM_DMABASE_CCR1, TIM_DMABASE_CCR2, TIM_DMABASE_CCR3 TIM_DMABASE_CCR4, TIM_DMABASE_BDTR, TIM_DMABASE_OR1, TIM_DMABASE_CCMR3, </pre>

	TIM_DMABASE_CCR5, TIM_DMABASE_CCR6, TIM_DMABASE_AF1, TIM_DMABASE_AF2, TIM_DMABASE_TISEL } uint32_t BurstRequestSrc: { TIM_DMA_UPDATE, TIM_DMA_CC1, TIM_DMA_CC2, TIM_DMA_CC3, TIM_DMA_CC4, TIM_DMA_COM, TIM_DMA_TRIGGER } uint32_t * BurstBuffer uint32_t BurstLength: [TIM_DMABURSTLENGTH_1TRANSFER, TIM_DMABURSTLENGTH_18TRANSFERS] uint32_t DataLength: [1, 0xFFFF]
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.15 HAL_TIM_DMABurst_ReadStop

3.23.15.1 功能介绍

停止 TIM DMA Burst 读取模式。

3.23.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_DMABurst_ReadStop (TIM_HandleTypeDef * htim, uint32_t BurstRequestSrc)
输入	TIM_HandleTypeDef * htim uint32_t BurstRequestSrc: { TIM_DMA_UPDATE, TIM_DMA_CC1, TIM_DMA_CC2, TIM_DMA_CC3, TIM_DMA_CC4, TIM_DMA_COM, TIM_DMA_TRIGGER }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.16 HAL_TIM_GenerateEvent

3.23.16.1 功能介绍

生成软件事件。

3.23.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_GenerateEvent (TIM_HandleTypeDef * htim, uint32_t EventSource)
输入	TIM_HandleTypeDef * htim uint32_t EventSource: { TIM_EVENTSOURCE_UPDATE, TIM_EVENTSOURCE_CC1, TIM_EVENTSOURCE_CC2, TIM_EVENTSOURCE_CC3,

	TIM_EVENTSOURCE_CC4, TIM_EVENTSOURCE_COM, TIM_EVENTSOURCE_TRIGGER, TIM_EVENTSOURCE_BREAK, TIM_EVENTSOURCE_BREAK2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.17 HAL_TIM_ReadCapturedValue

3.23.17.1 功能介绍

从捕获比较单元读取捕获的值。

3.23.17.2 接口定义

函数接口	uint32_t HAL_TIM_ReadCapturedValue (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim TIM_IC_InitTypeDef * sConfig uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.18 HAL_TIM_PeriodElapsedCallback

3.23.18.1 功能介绍

非阻塞模式下的周期运行回调。

3.23.18.2 接口定义

函数接口	void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.19 HAL_TIM_PeriodElapsedCallback

3.23.19.1 功能介绍

非阻塞模式下的周期运行一半回调。

3.23.19.2 接口定义

函数接口	void HAL_TIM_PeriodElapsedHalfCpltCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim

输出	无
返回值	无
资源使用	
说明	

3.23.20 HAL_TIM_OC_DelayElapsedCallback

3.23.20.1 功能介绍

非阻塞模式下的输出比较回调。

3.23.20.2 接口定义

函数接口	void HAL_TIM_OC_DelayElapsedCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.21 HAL_TIM_IC_CaptureCallback

3.23.21.1 功能介绍

非阻塞模式下的输入捕获回调。

3.23.21.2 接口定义

函数接口	void HAL_TIM_IC_CaptureCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.22 HAL_TIM_IC_CaptureHalfCpltCallback

3.23.22.1 功能介绍

非阻塞模式下的输入捕获完成一半回调。

3.23.22.2 接口定义

函数接口	void HAL_TIM_IC_CaptureHalfCpltCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.23 HAL_TIM_PWM_PulseFinishedCallback

3.23.23.1 功能介绍

在非阻塞模式 PWM 脉冲完成回调。

3.23.23.2 接口定义

函数接口	void HAL_TIM_PWM_PulseFinishedCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.24 HAL_TIM_PWM_PulseFinishedHalfCpltCallback

3.23.24.1 功能介绍

在非阻塞模式 PWM 脉冲完成一半回调。

3.23.24.2 接口定义

函数接口	void HAL_TIM_PWM_PulseFinishedHalfCpltCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.25 HAL_TIM_TriggerCallback

3.23.25.1 功能介绍

非阻塞模式下的触发检测回调。

3.23.25.2 接口定义

函数接口	void HAL_TIM_TriggerCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.26 HAL_TIM_TriggerHalfCpltCallback

3.23.26.1 功能介绍

非阻塞模式下的触发检测完成一半回调。

3.23.26.2 接口定义

函数接口	void HAL_TIM_TriggerHalfCpltCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim

输出	无
返回值	无
资源使用	
说明	

3.23.27 HAL_TIM_ErrorCallback

3.23.27.1 功能介绍

非阻塞模式下的错误回调。

3.23.27.2 接口定义

函数接口	void HAL_TIM_ErrorCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.28 HAL_TIM_Base_GetState

3.23.28.1 功能介绍

获取 TIM 时间基数状态。

3.23.28.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIM_Base_GetState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_StateTypeDef
资源使用	
说明	

3.23.29 HAL_TIM_OC_GetState

3.23.29.1 功能介绍

获取 TIM OC 状态。

3.23.29.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIM_OC_GetState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_StateTypeDef
资源使用	
说明	

3.23.30 HAL_TIM_PWM_GetState

3.23.30.1 功能介绍

获取 TIM PWM 状态。

3.23.30.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIM_PWM_GetState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_StateTypeDef
资源使用	
说明	

3.23.31 HAL_TIM_IC_GetState

3.23.31.1 功能介绍

获取 TIM IC 状态。

3.23.31.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIM_IC_GetState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_StateTypeDef
资源使用	
说明	

3.23.32 HAL_TIM_OnePulse_GetState

3.23.32.1 功能介绍

获取 TIM 单脉冲状态。

3.23.32.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIM_OnePulse_GetState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_StateTypeDef
资源使用	
说明	

3.23.33 HAL_TIM_Encoder_GetState

3.23.33.1 功能介绍

获取 TIM 编码器状态。

3.23.33.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIM_Encoder_GetState
------	---

	(TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_StateTypeDef
资源使用	
说明	

3.23.34 HAL_TIM_GetActiveChannel

3.23.34.1 功能介绍

获取 TIM 编码器模式状态。

3.23.34.2 接口定义

函数接口	HAL_TIM_ActiveChannel HAL_TIM_GetActiveChannel (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_TIM_ActiveChannel
资源使用	
说明	

3.23.35 HAL_TIM_GetChannelState

3.23.35.1 功能介绍

获取 TIM 通道的实际状态。

3.23.35.2 接口定义

函数接口	HAL_TIM_ChannelStateTypeDef HAL_TIM_GetChannelState (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_TIM_ChannelStateTypeDef
资源使用	
说明	

3.23.36 HAL_TIM_DMABurstState

3.23.36.1 功能介绍

获取 DMA Burst 操作的实际状态。。

3.23.36.2 接口定义

函数接口	HAL_TIM_DMABurstStateTypeDef HAL_TIM_DMABurstState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无

返回值	HAL_TIM_DMABurstStateTypeDef
资源使用	
说明	

3.23.37 HAL_TIM_Base_Init

3.23.37.1 功能介绍

初始化 TIM Base。

3.23.37.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Init (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.38 HAL_TIM_Base_DeInit

3.23.38.1 功能介绍

清除 TIM Base 初始化参数。

3.23.38.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.39 HAL_TIM_Base_MspInit

3.23.39.1 功能介绍

初始化 TIM Base MSP。

3.23.39.2 接口定义

函数接口	void HAL_TIM_Base_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.40 HAL_TIM_Base_MspDeInit

3.23.40.1 功能介绍

清除 TIM Base MSP 初始化参数。

3.23.40.2 接口定义

函数接口	void HAL_TIM_Base_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.41 HAL_TIM_Base_Start

3.23.41.1 功能介绍

开始生成 TIM Base。

3.23.41.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Start (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.42 HAL_TIM_Base_Stop

3.23.42.1 功能介绍

停止生成 TIM Base。

3.23.42.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Stop (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.43 HAL_TIM_Base_Start_IT

3.23.43.1 功能介绍

以中断模式开始生成 TIM Base。

3.23.43.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Start_IT (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.23.44 HAL_TIM_Base_Stop_IT

3.23.44.1 功能介绍

以中断模式停止生成 TIM Base。

3.23.44.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Stop_IT (TIM_HandleTypeDef * htim, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.45 HAL_TIM_Base_Start_DMA

3.23.45.1 功能介绍

以 DMA 模式开始生成 TIM Base。

3.23.45.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Start_DMA(TIM_HandleTypeDef * htim, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.46 HAL_TIM_Base_Stop_DMA

3.23.46.1 功能介绍

以 DMA 模式停止生成 TIM Base。

3.23.46.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Base_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.47 HAL_TIM_OC_Init

3.23.47.1 功能介绍

初始化 TIM OC。

3.23.47.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Init (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.48 HAL_TIM_OC_DeInit

3.23.48.1 功能介绍

清除 TIM OC 初始化参数。

3.23.48.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.49 HAL_TIM_OC_MspInit

3.23.49.1 功能介绍

初始化 TIM OC MSP。

3.23.49.2 接口定义

函数接口	void HAL_TIM_OC_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.50 HAL_TIM_OC_MspDeInit

3.23.50.1 功能介绍

清除 TIM OC MSP 初始化参数。

3.23.50.2 接口定义

函数接口	void HAL_TIM_OC_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim

输出	无
返回值	无
资源使用	
说明	

3.23.51 HAL_TIM_OC_Start

3.23.51.1 功能介绍

开始生成 TIM OC。

3.23.51.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.52 HAL_TIM_OC_Stop

3.23.52.1 功能介绍

停止生成 TIM OC。

3.23.52.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.53 HAL_TIM_OC_Start_IT

3.23.53.1 功能介绍

以中断模式开始生成 TIM OC。

3.23.53.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel:

	{ TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.54 HAL_TIM_OC_Stop_IT

3.23.54.1 功能介绍

以中断模式停止生成 TIM OC。

3.23.54.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.55 HAL_TIM_OC_Start_DMA

3.23.55.1 功能介绍

以 DMA 模式开始生成 TIM OC。

3.23.55.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Start_DMA(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 } uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.56 HAL_TIM_OC_Stop_DMA

3.23.56.1 功能介绍

以 DMA 模式停止生成 TIM OC。

3.23.56.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OC_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.57 HAL_TIM_PWM_Init

3.23.57.1 功能介绍

初始化 TIM PWM。

3.23.57.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Init (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.58 HAL_TIM_PWM_DeInit

3.23.58.1 功能介绍

清除 TIM OC 初始化参数。

3.23.58.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.59 HAL_TIM_PWM_MspInit

3.23.59.1 功能介绍

初始化 TIM PWM MSP。

3.23.59.2 接口定义

函数接口	void HAL_TIM_PWM_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim

输出	无
返回值	无
资源使用	
说明	

3.23.60 HAL_TIM_PWM_MspDeInit

3.23.60.1 功能介绍

清除 TIM PWM MSP 初始化参数。

3.23.60.2 接口定义

函数接口	void HAL_TIM_PWM_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.61 HAL_TIM_PWM_Start

3.23.61.1 功能介绍

开始生成 TIM PWM。

3.23.61.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.62 HAL_TIM_PWM_Stop

3.23.62.1 功能介绍

停止生成 TIM PWM。

3.23.62.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4, TIM_CHANNEL_5, TIM_CHANNEL_6 }
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.63 HAL_TIM_PWM_Start_IT

3.23.63.1 功能介绍

以中断模式开始生成 TIM PWM。

3.23.63.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.64 HAL_TIM_PWM_Stop_IT

3.23.64.1 功能介绍

以中断模式停止生成 TIM Base。

3.23.64.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.65 HAL_TIM_PWM_Start_DMA

3.23.65.1 功能介绍

以 DMA 模式开始生成 TIM PWM。

3.23.65.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Start_DMA(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3,

	TIM_CHANNEL_4 } uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.66 HAL_TIM_PWM_Stop_DMA

3.23.66.1 功能介绍

以 DMA 模式停止生成 TIM PWM。

3.23.66.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_PWM_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.67 HAL_TIM_IC_Init

3.23.67.1 功能介绍

初始化 TIM IC。

3.23.67.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Init (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.68 HAL_TIM_IC_DeInit

3.23.68.1 功能介绍

清除 TIM IC 初始化参数。

3.23.68.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.69 HAL_TIM_IC_MspInit

3.23.69.1 功能介绍

初始化 TIM IC MSP。

3.23.69.2 接口定义

函数接口	void HAL_TIM_IC_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.70 HAL_TIM_IC_MspDeInit

3.23.70.1 功能介绍

清除 TIM IC MSP 初始化参数。

3.23.70.2 接口定义

函数接口	void HAL_TIM_IC_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.71 HAL_TIM_IC_Start

3.23.71.1 功能介绍

开始生成 TIM IC。

3.23.71.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.72 HAL_TIM_IC_Stop

3.23.72.1 功能介绍

停止生成 TIM IC。

3.23.72.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.73 HAL_TIM_IC_Start_IT

3.23.73.1 功能介绍

以中断模式开始生成 TIM IC。

3.23.73.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.74 HAL_TIM_IC_Stop_IT

3.23.74.1 功能介绍

以中断模式停止生成 TIM IC。

3.23.74.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.23.75 HAL_TIM_IC_Start_DMA

3.23.75.1 功能介绍

以 DMA 模式开始生成 TIM IC。

3.23.75.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Start_DMA(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 } uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.76 HAL_TIM_IC_Stop_DMA

3.23.76.1 功能介绍

以 DMA 模式停止生成 TIM IC。

3.23.76.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_IC_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.77 HAL_TIM_OnePulse_Init

3.23.77.1 功能介绍

初始化 TIM 单脉冲模式。

3.23.77.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_Init (TIM_HandleTypeDef * htim, uint32_t OnePulseMode)
输入	TIM_HandleTypeDef * htim uint32_t OnePulseMode:

	{TIM_OPMODE_SINGLE, TIM_OPMODE_REPETITIVE}
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.78 HAL_TIM_OnePulse_DeInit

3.23.78.1 功能介绍

清除 TIM 单脉冲模式初始化参数。

3.23.78.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.79 HAL_TIM_OnePulse_MspInit

3.23.79.1 功能介绍

初始化 TIM 单脉冲模式 MSP。

3.23.79.2 接口定义

函数接口	void HAL_TIM_OnePulse_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.80 HAL_TIM_OnePulse_MspDeInit

3.23.80.1 功能介绍

清除 TIM 单脉冲模式 MSP 初始化参数。

3.23.80.2 接口定义

函数接口	void HAL_TIM_OnePulse_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.81 HAL_TIM_OnePulse_Start

3.23.81.1 功能介绍

开始生成 TIM 单脉冲模式。

3.23.81.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_Start (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
输入	TIM_HandleTypeDef * htim uint32_t OutputChannel
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.82 HAL_TIM_OnePulse_Stop

3.23.82.1 功能介绍

停止生成 TIM 单脉冲模式。

3.23.82.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_Stop(TIM_HandleTypeDef * htim, uint32_t OutputChannel)
输入	TIM_HandleTypeDef * htim uint32_t OutputChannel
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.83 HAL_TIM_OnePulse_Start_IT

3.23.83.1 功能介绍

以中断模式开始生成 TIM 单脉冲模式。

3.23.83.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_Start_IT (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
输入	TIM_HandleTypeDef * htim uint32_t OutputChannel
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.84 HAL_TIM_OnePulse_Stop_IT

3.23.84.1 功能介绍

以中断模式停止生成 TIM 单脉冲模式。

3.23.84.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_OnePulse_Stop_IT (TIM_HandleTypeDef * htim, uint32_t OutputChannel)
输入	TIM_HandleTypeDef * htim uint32_t OutputChannel
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.85 HAL_TIM_Encoder_Init

3.23.85.1 功能介绍

初始化 TIM Encoder。

3.23.85.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Init (TIM_HandleTypeDef * htim, TIM_Encoder_InitTypeDef * sConfig)
输入	TIM_HandleTypeDef * htim TIM_Encoder_InitTypeDef * sConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.86 HAL_TIM_Encoder_DeInit

3.23.86.1 功能介绍

清除 TIM Encoder 初始化参数。

3.23.86.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.87 HAL_TIM_Encoder_MspInit

3.23.87.1 功能介绍

初始化 TIM Encoder MSP。

3.23.87.2 接口定义

函数接口	void HAL_TIM_Encoder_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无

返回值	无
资源使用	
说明	

3.23.88 HAL_TIM_Encoder_MspDeInit

3.23.88.1 功能介绍

清除 TIM Encoder MSP 初始化参数。

3.23.88.2 接口定义

函数接口	void HAL_TIM_Encoder_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.89 HAL_TIM_Encoder_Start

3.23.89.1 功能介绍

开始生成 TIM Encoder。

3.23.89.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_ALL }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.90 HAL_TIM_Encoder_Stop

3.23.90.1 功能介绍

停止生成 TIM Encoder。

3.23.90.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Stop (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_ALL }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.91 HAL_TIM_Encoder_Start_IT

3.23.91.1 功能介绍

以中断模式开始生成 TIM Encoder。

3.23.91.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_ALL }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.92 HAL_TIM_Encoder_Stop_IT

3.23.92.1 功能介绍

以中断模式停止生成 TIM Encoder。

3.23.92.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_ALL }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.93 HAL_TIM_Encoder_Start_DMA

3.23.93.1 功能介绍

以 DMA 模式开始生成 TIM Encoder。

3.23.93.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Start_DMA(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData1, uint32_t * pData2, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_ALL } uint32_t * pData1 uint32_t * pData2 uint16_t Length
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.94 HAL_TIM_Encoder_Stop_DMA

3.23.94.1 功能介绍

以 DMA 模式停止生成 TIM Encoder。

3.23.94.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIM_Encoder_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.95 HAL_TIM_IRQHandler

3.23.95.1 功能介绍

处理 TIM 中断请求。

3.23.95.2 接口定义

函数接口	void HAL_TIM_IRQHandler (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_ALL }
输出	无
返回值	无
资源使用	
说明	

3.23.96 HAL_TIMEx_HallSensor_Init

3.23.96.1 功能介绍

初始化 TIM 霍尔传感器接口。

3.23.96.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Init (TIM_HandleTypeDef * htim, TIM_HallSensor_InitTypeDef * sConfig)
输入	TIM_HandleTypeDef * htim TIM_HallSensor_InitTypeDef * sConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.97 HAL_TIMEx_HallSensor_DeInit

3.23.97.1 功能介绍

清除 TIM 霍尔传感器接口初始化参数。

3.23.97.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_DeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.98 HAL_TIMEx_HallSensor_MspInit

3.23.98.1 功能介绍

初始化 TIM 霍尔传感器接口 MSP。

3.23.98.2 接口定义

函数接口	void HAL_TIMEx_HallSensor_MspInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.99 HAL_TIMEx_HallSensor_MspDeInit

3.23.99.1 功能介绍

清除 TIM 霍尔传感器接口 MSP 初始化参数。

3.23.99.2 接口定义

函数接口	void HAL_TIMEx_HallSensor_MspDeInit (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	无
资源使用	
说明	

3.23.100 HAL_TIMEx_HallSensor_Start

3.23.100.1 功能介绍

开始生成 TIM 霍尔传感器接口。

3.23.100.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Start (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.101 HAL_TIMEx_HallSensor_Stop

3.23.101.1 功能介绍

停止生成 TIM 霍尔传感器接口。

3.23.101.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Stop(TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.102 HAL_TIMEx_HallSensor_Start_IT

3.23.102.1 功能介绍

以中断模式开始生成 TIM 霍尔传感器接口。

3.23.102.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Start_IT (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.103 HAL_TIMEx_HallSensor_Stop_IT

3.23.103.1 功能介绍

以中断模式停止生成 TIM 霍尔传感器接口。

3.23.103.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Stop_IT (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.104 HAL_TIMEx_HallSensor_Start_DMA

3.23.104.1 功能介绍

以 DMA 模式开始生成 TIM 霍尔传感器接口。

3.23.104.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Start_DMA(TIM_HandleTypeDef * htim, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.105 HAL_TIMEx_HallSensor_Stop_DMA

3.23.105.1 功能介绍

以 DMA 模式停止生成 TIM 霍尔传感器接口。

3.23.105.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_HallSensor_Stop_DMA (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.106 HAL_TIMEx_OCN_Start

3.23.106.1 功能介绍

启动 TIM 输出在互补输出上产生比较信号。

3.23.106.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OCN_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.107 HAL_TIMEx_OCN_Stop

3.23.107.1 功能介绍

停止 TIM 输出在互补输出上产生比较信号。

3.23.107.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OCN_Stop(TIM_HandleTypeDef * htim, htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.108 HAL_TIMEx_OCN_Start_IT

3.23.108.1 功能介绍

以中断模式开始生成 TIM 输出在互补输出上产生比较信号。

3.23.108.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OCN_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.109 HAL_TIMEx_OCN_Stop_IT

3.23.109.1 功能介绍

以中断模式停止生成 TIM 输出在互补输出上产生比较信号。

3.23.109.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OCN_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.110 HAL_TIMEx_OCN_Start_DMA

3.23.110.1 功能介绍

以 DMA 模式开始生成 TIM 输出在互补输出上产生比较信号。

3.23.110.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OCN_Start_DMA(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 } uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.111 HAL_TIMEx_OCN_Stop_DMA

3.23.111.1 功能介绍

以 DMA 模式停止生成 TIM 输出在互补输出上产生比较信号。

3.23.111.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OCN_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.112 HAL_TIMEx_PWMN_Start

3.23.112.1 功能介绍

启动 TIM 输出在互补输出上产生 PWM 信号。

3.23.112.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_PWMN_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.23.113 HAL_TIMEx_PWMN_Stop

3.23.113.1 功能介绍

停止 TIM 输出在互补输出上产生 PWM 信号。

3.23.113.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_PWMN_Stop(TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.114 HAL_TIMEx_PWMN_Start_IT

3.23.114.1 功能介绍

以中断模式开始生成 TIM 输出在互补输出上产生 PWM 信号。

3.23.114.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_PWMN_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.115 HAL_TIMEx_PWMN_Stop_IT

3.23.115.1 功能介绍

以中断模式停止生成 TIM 输出在互补输出上产生 PWM 信号。

3.23.115.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_PWMN_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.23.116 HAL_TIMEx_PWMN_Start_DMA

3.23.116.1 功能介绍

以 DMA 模式开始生成 TIM 输出在互补输出上产生 PWM 信号。

3.23.116.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_PWMN_Start_DMA(TIM_HandleTypeDef * htim, uint32_t Channel, uint32_t * pData, uint16_t Length)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 } uint32_t * pData uint16_t Length
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.117 HAL_TIMEx_PWMN_Stop_DMA

3.23.117.1 功能介绍

以 DMA 模式停止生成 TIM 输出在互补输出上产生 PWM 信号。

3.23.117.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_PWMN_Stop_DMA (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.118 HAL_TIMEx_OnePulseN_Start

3.23.118.1 功能介绍

启动 TIM 输出在互补输出上产生单脉冲信号。

3.23.118.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Start (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2 }

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.119 HAL_TIMEx_OnePulseN_Stop

3.23.119.1 功能介绍

停止 TIM 输出在互补输出上产生单脉冲信号。

3.23.119.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Stop(TIM_HandleTypeDef * htim, htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.120 HAL_TIMEx_OnePulseN_Start_IT

3.23.120.1 功能介绍

以中断模式开始生成 TIM 输出在互补输出上产生单脉冲信号。

3.23.120.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Start_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.121 HAL_TIMEx_OnePulseN_Stop_IT

3.23.121.1 功能介绍

以中断模式停止生成 TIM 输出在互补输出上产生单脉冲信号。

3.23.121.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_OnePulseN_Stop_IT (TIM_HandleTypeDef * htim, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t Channel: { TIM_CHANNEL_1, TIM_CHANNEL_2 }

输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.122 HAL_TIMEx_ConfigCommutEvent

3.23.122.1 功能介绍

配置 TIM 对换事件顺序。

3.23.122.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_ConfigCommutEvent (TIM_HandleTypeDef * htim, uint32_t InputTrigger, uint32_t CommutationSource)
输入	TIM_HandleTypeDef * htim uint32_t InputTrigger: { TIM_TS_ITR0, TIM_TS_ITR1, TIM_TS_ITR2, TIM_TS_ITR3, TIM_TS_NONE } uint32_t CommutationSource: { TIM_COMMUTATION_TRGI, TIM_COMMUTATION_SOFTWARE }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.123 HAL_TIMEx_ConfigCommutEvent_IT

3.23.123.1 功能介绍

以中断模式配置 TIM 对换事件顺序。

3.23.123.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_ConfigCommutEvent_IT (TIM_HandleTypeDef * htim, uint32_t InputTrigger, uint32_t CommutationSource)
输入	TIM_HandleTypeDef * htim uint32_t InputTrigger: { TIM_TS_ITR0, TIM_TS_ITR1, TIM_TS_ITR2, TIM_TS_ITR3, TIM_TS_NONE } uint32_t CommutationSource: { TIM_COMMUTATION_TRGI, TIM_COMMUTATION_SOFTWARE }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.124 HAL_TIMEx_ConfigCommutEvent_DMA

3.23.124.1 功能介绍

以 DMA 模式配置 TIM 对换事件顺序。

3.23.124.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_ConfigCommutEvent_DMA (TIM_HandleTypeDef * htim, uint32_t InputTrigger, uint32_t CommutationSource)
输入	TIM_HandleTypeDef * htim uint32_t InputTrigger: { TIM_TS_ITR0, TIM_TS_ITR1, TIM_TS_ITR2, TIM_TS_ITR3, TIM_TS_NONE } uint32_t CommutationSource: { TIM_COMMUTATION_TRGI, TIM_COMMUTATION_SOFTWARE }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.125 HAL_TIMEx_MasterConfigSynchronization

3.23.125.1 功能介绍

配置 TIM 为主模式。

3.23.125.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_MasterConfigSynchronization (TIM_HandleTypeDef * htim, TIM_MasterConfigTypeDef * sMasterConfig)
输入	TIM_HandleTypeDef * htim TIM_MasterConfigTypeDef * sMasterConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.126 HAL_TIMEx_ConfigBreakDeadTime

3.23.126.1 功能介绍

配置断开功能，死区时间，锁级别，OSSI/OSSR 状态和 AOE(自动输出启用)。。

3.23.126.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_ConfigBreakDeadTime (TIM_HandleTypeDef * htim, TIM_BreakDeadTimeConfigTypeDef * sBreakDeadTimeConfig)
输入	TIM_HandleTypeDef * htim TIM_BreakDeadTimeConfigTypeDef * sBreakDeadTimeConfig
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.127 HAL_TIMEx_ConfigBreakInput

3.23.127.1 功能介绍

配置中断输入源。

3.23.127.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_ConfigBreakInput (TIM_HandleTypeDef * htim, uint32_t BreakInput, TIMEx_BreakInputConfigTypeDef * sBreakInputConfig)
输入	TIM_HandleTypeDef * htim TIMEx_BreakInputConfigTypeDef * sBreakInputConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.128 HAL_TIMEx_GroupChannel5

3.23.128.1 功能介绍

通道 5 和通道 1、2 或 3 组合。

3.23.128.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_GroupChannel5 (TIM_HandleTypeDef * htim, uint32_t Channels)
输入	TIM_HandleTypeDef * htim uint32_t Channels
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.129 HAL_TIMEx_RemapConfig

3.23.129.1 功能介绍

配置 TIMx 重映射输入功能。

3.23.129.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_RemapConfig (TIM_HandleTypeDef * htim, uint32_t Remap)
输入	TIM_HandleTypeDef * htim uint32_t Remap: { TIM1: TIM_TIM1_ETR_GPIO, TIM_TIM1_ETR_COMP1, TIM_TIM1_ETR_COMP2, TIM_TIM1_ETR_COMP3, TIM_TIM1_ETR_COMP4, TIM_TIM1_ETR_ADC1_AWD1

	TIM2: TIM_TIM2_ETR_GPIO, TIM_TIM2_ETR_COMP1, TIM_TIM2_ETR_COMP2, TIM_TIM2_ETR_LSE TIM3: TIM_TIM3_ETR_GPIO, TIM_TIM3_ETR_COMP1, TIM_TIM3_ETR_COMP2, TIM_TIM2_ETR_LSE TIM4: TIM_TIM4_ETR_COMP1, TIM_TIM4_ETR_COMP2, TIM_TIM2_ETR_LSE }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.130 HAL_TIMEx_TISelection

3.23.130.1 功能介绍

选择定时器输入源。

3.23.130.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_TISelection (TIM_HandleTypeDef * htim, uint32_t TISelection, uint32_t Channel)
输入	TIM_HandleTypeDef * htim uint32_t TISelection uint32_t Channels: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3, TIM_CHANNEL_4 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.131 HAL_TIMEx_DisarmBreakInput

3.23.131.1 功能介绍

解除指定的断路器输入。

3.23.131.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_DisarmBreakInput (TIM_HandleTypeDef * htim, uint32_t BreakInput)
输入	TIM_HandleTypeDef * htim uint32_t BreakInput: { TIM_BREAKINPUT_BRK, TIM_BREAKINPUT_BRK2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.132 HAL_TIMEx_ReArmBreakInput

3.23.132.1 功能介绍

设置指定的断路器输入。

3.23.132.2 接口定义

函数接口	HAL_StatusTypeDef HAL_TIMEx_ReArmBreakInput (TIM_HandleTypeDef * htim, uint32_t BreakInput)
输入	TIM_HandleTypeDef * htim uint32_t BreakInput: { TIM_BREAKINPUT_BRK, TIM_BREAKINPUT_BRK2 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.133 HAL_TIMEx_CommutCallback

3.23.133.1 功能介绍

在非阻塞模式下霍尔换向回调。

3.23.133.2 接口定义

函数接口	void HAL_TIMEx_CommutCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.134 HAL_TIMEx_CommutHalfCpltCallback

3.23.134.1 功能介绍

在非阻塞模式下霍尔换向一半回调。

3.23.134.2 接口定义

函数接口	void Hall commutation changed half complete callback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.135 HAL_TIMEx_BreakCallback

3.23.135.1 功能介绍

在非阻塞模式下霍尔断路检测回调。

3.23.135.2 接口定义

函数接口	void HAL_TIMEx_BreakCallback (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.136 HAL_TIMEx_BreakCallback2

3.23.136.1 功能介绍

在非阻塞模式下霍尔断路 2 检测回调。

3.23.136.2 接口定义

函数接口	void HAL_TIMEx_BreakCallback2 (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.137 HAL_TIMEx_HallSensor_GetState

3.23.137.1 功能介绍

获取 TIM 霍尔传感器接口状态。

3.23.137.2 接口定义

函数接口	HAL_TIM_StateTypeDef HAL_TIMEx_HallSensor_GetState (TIM_HandleTypeDef * htim)
输入	TIM_HandleTypeDef * htim
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.23.138 HAL_TIMEx_GetChannelNState

3.23.138.1 功能介绍

获取 TIM 霍尔传感器通道状态。

3.23.138.2 接口定义

函数接口	HAL_TIM_ChannelStateTypeDef HAL_TIMEx_GetChannelNState (TIM_HandleTypeDef * htim, uint32_t ChannelN)
输入	TIM_HandleTypeDef * htim uint32_t ChannelN: { TIM_CHANNEL_1, TIM_CHANNEL_2, TIM_CHANNEL_3 }
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24 UART 模块

属性	类型	字段名	含义
USART_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	CR3	控制寄存器 3
读写	uint32_t	BRR	波特率分频寄存器
读写	uint32_t	GTPR	保护时间和分频寄存器
读写	uint32_t	RTOR	超时块传输长度寄存器
只写	uint32_t	RQR	请求寄存器

只读	uint32_t	ISR	中断状态寄存器
只写	uint32_t	ICR	中断标志清零寄存器
只读	uint32_t	RDR	接收数据寄存器
读写	uint32_t	TDR	发送数据寄存器
读写	uint32_t	PRESC	预分频寄存器
UART_InitTypeDef			
读写	uint32_t	BaudRate	波特率
读写	uint32_t	WordLength	字符长度
读写	uint32_t	StopBits	停止位
读写	uint32_t	Parity	RS485收发器使能信号极性
读	uint32_t	Mode	模式

写			
读写	uint32_t	HwFlowCtl	硬件流控制
读写	uint32_t	OverSampling	过采样模式
读写	uint32_t	OneBitSampling	单采样
读写	uint32_t	ClockPrescaler	时钟分频值
UART_AdvFeatureInitTypeDef			
读写	uint32_t	AdvFeatureInit	高级功能初始化
读写	uint32_t	TxPinLevelInvert	TX 引脚激电平反转
读写	uint32_t	RxPinLevelInvert	RX 引脚激电平反转
读写	uint32_t	DataInvert	数据反转
读写	uint32_t	Swap	交换
读写	uint32_t	OverrunDisable	禁用接收超限检测

读 写	uint32_t	DMADisableonRxError	接 收 错 误 时 禁 用 DMA
读 写	uint32_t	AutoBaudRateEnable	使 能 特 自 检 测
读 写	uint32_t	AutoBaudRateMode	波 特 率 自 检 测 模 式
读 写	uint32_t	MSBFirst	先 发 送 MSB
UART_HandleTypeDef			
读 写	USART_TypeDef	Instance	寄 存 器 地 址
读 写	UART_InitTypeDef	Init	初 始 参 数
读 写	UART_AdvFeatureInitTypeDef	AdvancedInit	高 级 功 能 初 始 参 数
读 写	uint8_t	pTxBuffPtr	TX 传 输 缓 冲 区 的 指 针

读 写	uint16_t	TxXferSize	传 输 发 送 大 小
读 写	uint16_t	TxXferCount	传 输 发 送 计 数 器
读 写	uint16_t	pRxBuffPtr	RX 传 输 缓 冲 区 的 指 针
读 写	uint16_t	RxXferSize	传 输 接 收 大 小
读 写	uint16_t	RxXferCount	传 输 接 收 计 数 器
读 写	uint16_t	Mask	UART Rx RDR 注 册 掩 码
读 写	uint16_t	NbRxDataToProcess	在 RX ISR 执 行 期 间 要 处 理 的 数 据 数
读 写	uint16_t	NbTxDataToProcess	在 TX ISR 执 行 期 间 要

			处 理 的 数 据 数
读 写	uint32_t	FifoMode	FIFO 模式
读 写	HAL_UART_RxTypeTypeDef	ReceptionType	持 续 接 收 类 型
读 写	uint16_t	NbRxDataToProcess	在 RX ISR 执 行 期 间 要 处 理 的 数 据 数
读 写	uint16_t	NbTxDataToProcess	在 TX ISR 执 行 期 间 要 处 理 的 数 据 数
读 写	指针函数	void(*RxISR)(struct __UART_HandleTypeDef *huart)	Rx ISR 函 数 指 针
读 写	指针函数	void(*TxISR)(struct __UART_HandleTypeDef *huart)	Tx ISR 函 数 指 针
读 写	DMA_HandleTypeDef	hdmatx	DMA 句柄
读 写	DMA_HandleTypeDef	hdmarx	DMA 句柄
读 写	HAL_LockTypeDef	Lock	锁 定

读 写	HAL_UART_StateTypeDef	gState	UART 状态
读 写	HAL_SMARTCARD_StateTypeDef	RxState	UART 接收 状态
读 写	uint32_t	ErrorCode	错 误 代 码
读 写	指针函数	void(*TxHalfCpltCallback)(struct __UART_HandleTypeDef *huart)	发 送 传 输 完 成 一 半 回 调
读 写	指针函数	void(*TxCpltCallback)(struct __UART_HandleTypeDef *huart)	发 送 传 输 完 成 回 调
读 写	指针函数	void(*RxHalfCpltCallback)(struct __UART_HandleTypeDef *huart)	接 收 传 输 完 成 一 半 回 调
读 写	指针函数	void(*RxCpltCallback)(struct __UART_HandleTypeDef *huart)	接 收 传 输 完 成 回 调
读 写	指针函数	void(*ErrorCallback)(struct __UART_HandleTypeDef *huart)	错 误 回 调
读 写	指针函数	void(*AbortCpltCallback)(struct __UART_HandleTypeDef *huart)	中 止 完 成 回 调
读 写	指针函数	void(*AbortTransmitCpltCallback)(struct __UART_HandleTypeDef *huart)	发 送 中 止 完 成 回 调
读	指针函数	void(*AbortReceiveCpltCallback)(struct	接 收

写		__UART_HandleTypeDef *huart)	中 止 完 成 回 调
读 写	指针函数	void(*WakeupCallback)(struct __UART_HandleTypeDef *huart)	唤 醒 完 成 回 调
读 写	指针函数	void(*RxFifoFullCallback)(struct __UART_HandleTypeDef *huart)	接 收 FIFO 满 回 调
读 写	指针函数	void(*TxFifoEmptyCallback)(struct __UART_HandleTypeDef *huart)	发 送 FIFO 空 回 调
读 写	指针函数	void(*RxEventCallback)(struct __UART_HandleTypeDef*huart, uint16_t Pos)	接 收 事 件 回 调
读 写	指针函数	voi(*MspInitCallback)(struct __UART_HandleTypeDef *huart)	MSP 初 始 化 回 调
读 写	指针函数	void(*MspDeInitCallback)(struct __UART_HandleTypeDef *huart)	MSP 非 初 始 化 回 调

3.24.1 HAL_UART_Init

3.24.1.1 功能介绍

初始化 UART。

3.24.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Init (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.2 HAL_HalfDuplex_Init

3.24.2.1 功能介绍

初始化 UART 半双工模式。

3.24.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_HalfDuplex_Init (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.3 HAL_MultiProcessor_Init

3.24.3.1 功能介绍

初始化 UART 多处理器模式。

3.24.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_MultiProcessor_Init (UART_HandleTypeDef * huart, uint8_t Address, uint32_t WakeUpMethod)
输入	UART_HandleTypeDef * huart uint8_t Address uint32_t WakeUpMethod: { UART_WAKEUPMETHOD_ADDRESSMARK, UART_WAKEUPMETHOD_IDLELINE }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.4 HAL_UART_DeInit

3.24.4.1 功能介绍

清除 UART 初始化参数。

3.24.4.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_DeInit (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.5 HAL_UART_MspInit

3.24.5.1 功能介绍

UART MSP 初始化。

3.24.5.2 接口定义

函数接口	void HAL_UART_MspInit (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.6 HAL_UART_MspDeInit

3.24.6.1 功能介绍

清除 UART MSP 初始化参数。

3.24.6.2 接口定义

函数接口	void HAL_UART_MspDeInit (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.7 HAL_UART_Transmit

3.24.7.1 功能介绍

UART 在阻塞方式发送数据。

3.24.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Transmit (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.8 HAL_UART_Receive

3.24.8.1 功能介绍

UART 在阻塞方式接收数据。

3.24.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Receive (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint32_t Timeout)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.9 HAL_UART_Transmit_IT

3.24.9.1 功能介绍

UART 以中断模式在阻塞方式发送数据。

3.24.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Transmit_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.10 HAL_UART_Receive_IT

3.24.10.1 功能介绍

UART 以中断模式在阻塞方式接收数据。

3.24.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Receive_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.11 HAL_UART_Transmit_DMA

3.24.11.1 功能介绍

UART 以 DMA 模式在阻塞方式发送数据。

3.24.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Transmit_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.12 HAL_UART_Receive_DMA

3.24.12.1 功能介绍

UART 以 DMA 模式在阻塞方式接收数据。

3.24.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Receive_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.13 HAL_UART_DMAPause

3.24.13.1 功能介绍

暂停 DMA 传输。

3.24.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_DMAPause (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.14 HAL_UART_DMAResume

3.24.14.1 功能介绍

恢复 DMA 传输。

3.24.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_DMAResume (UART_HandleTypeDef
------	---

	* huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.15 HAL_UART_DMAStop

3.24.15.1 功能介绍

停止 DMA 传输。

3.24.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_DMAStop (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.16 HAL_UART_Abort

3.24.16.1 功能介绍

中止正在进行的传输(阻塞模式)。

3.24.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Abort (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.17 HAL_UART_AbortTransmit

3.24.17.1 功能介绍

中止正在进行的发送(阻塞模式)。

3.24.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_AbortTransmit (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.24.18 HAL_UART_AbortReceive

3.24.18.1 功能介绍

中止正在进行的接收(阻塞模式)。

3.24.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_AbortReceive (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.19 HAL_UART_Abort_IT

3.24.19.1 功能介绍

以中断模式中中止正在进行的传输(阻塞模式)。

3.24.19.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_Abort_IT (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.20 HAL_UART_AbortTransmit_IT

3.24.20.1 功能介绍

以中断模式中中止正在进行的发送(阻塞模式)。

3.24.20.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_AbortTransmit_IT (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.21 HAL_UART_AbortReceive_IT

3.24.21.1 功能介绍

以中断模式中中止正在进行的接收(阻塞模式)。

3.24.21.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_AbortReceive_IT (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.22 HAL_UART_IRQHandler

3.24.22.1 功能介绍

处理 UART 的中断请求。

3.24.22.2 接口定义

函数接口	void HAL_UART_IRQHandler (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.23 HAL_UART_TxHalfCpltCallback

3.24.23.1 功能介绍

UART 发送完成一半回调。

3.24.23.2 接口定义

函数接口	void HAL_UART_TxHalfCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.24 HAL_UART_TxCpltCallback

3.24.24.1 功能介绍

UART 发送完成回调。

3.24.24.2 接口定义

函数接口	void HAL_UART_TxCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.25 HAL_UART_RxHalfCpltCallback

3.24.25.1 功能介绍

UART 接收完成一半回调。

3.24.25.2 接口定义

函数接口	void HAL_UART_RxHalfCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.26 HAL_UART_RxCpltCallback

3.24.26.1 功能介绍

UART 接收完成回调。

3.24.26.2 接口定义

函数接口	void HAL_UART_RxCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.27 HAL_UART_ErrorCallback

3.24.27.1 功能介绍

UART 错误回调。

3.24.27.2 接口定义

函数接口	void HAL_UART_ErrorCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.28 HAL_UART_AbortCpltCallback

3.24.28.1 功能介绍

UART 中止完成回调。

3.24.28.2 接口定义

函数接口	void HAL_UART_AbortCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无

返回值	无
资源使用	
说明	

3.24.29 HAL_UART_AbortTransmitCpltCallback

3.24.29.1 功能介绍

UART 中止发送完成回调。

3.24.29.2 接口定义

函数接口	void HAL_UART_AbortTransmitCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.30 HAL_UART_AbortReceiveCpltCallback

3.24.30.1 功能介绍

UART 中止接收完成回调。

3.24.30.2 接口定义

函数接口	void HAL_UART_AbortReceiveCpltCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.31 HAL_UART_ReceiverTimeout_Config

3.24.31.1 功能介绍

动态更新 RTOR 寄存器中的接收器超时值。

3.24.31.2 接口定义

函数接口	void HAL_UART_ReceiverTimeout_Config (UART_HandleTypeDef * huart, uint32_t TimeoutValue)
输入	UART_HandleTypeDef * huart uint32_t TimeoutValue: [0x0, 0xFFFFFFFF]
输出	无
返回值	无
资源使用	
说明	

3.24.32 HAL_UART_EnableReceiverTimeout

3.24.32.1 功能介绍

使能 UART 接收器超时功能。

3.24.32.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_EnableReceiverTimeout (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.33 HAL_UART_DisableReceiverTimeout

3.24.33.1 功能介绍

禁止使能 UART 接收器超时功能。

3.24.33.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UART_DisableReceiverTimeout (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.34 HAL_MultiProcessor_EnableMuteMode

3.24.34.1 功能介绍

在静音模式下使能 UART (并不意味着 UART 进入静音模式)。

3.24.34.2 接口定义

函数接口	HAL_StatusTypeDef HAL_MultiProcessor_EnableMuteMode (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.35 HAL_MultiProcessor_DisableMuteMode

3.24.35.1 功能介绍

在静音模式下禁止使能 UART (并不意味着 UART 实际上退出静音模式, 因为它可能没有在这个时刻处于静音模式)。

3.24.35.2 接口定义

函数接口	HAL_StatusTypeDef HAL_MultiProcessor_DisableMuteMode (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.36 HAL_MultiProcessor_EnterMuteMode

3.24.36.1 功能介绍

进入 UART 静音模式(即 UART 实际进入静音模式)。

3.24.36.2 接口定义

函数接口	void HAL_MultiProcessor_EnterMuteMode (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.37 HAL_HalfDuplex_EnableTransmitter

3.24.37.1 功能介绍

启用 UART 发射器，禁用 UART 接收器。

3.24.37.2 接口定义

函数接口	HAL_StatusTypeDef HAL_HalfDuplex_EnableTransmitter (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.38 HAL_HalfDuplex_EnableReceiver

3.24.38.1 功能介绍

启用 UART 接收器，禁用 UART 发射器。

3.24.38.2 接口定义

函数接口	HAL_StatusTypeDef HAL_HalfDuplex_EnableReceiver (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无

返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.39 HAL_UART_GetState

3.24.39.1 功能介绍

获取 UART 状态。

3.24.39.2 接口定义

函数接口	HAL_UART_StateTypeDef HAL_UART_GetState (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_UART_StateTypeDef
资源使用	
说明	

3.24.40 HAL_UART_GetError

3.24.40.1 功能介绍

获取 UART 错误。

3.24.40.2 接口定义

函数接口	uint32_t HAL_UART_GetError (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	错误值
资源使用	
说明	

3.24.41 HAL_RS485Ex_Init

3.24.41.1 功能介绍

初始化 RS485 驱动。

3.24.41.2 接口定义

函数接口	HAL_StatusTypeDef HAL_RS485Ex_Init (UART_HandleTypeDef * huart, uint32_t Polarity, uint32_t AssertionTime, uint32_t DeassertionTime)
输入	UART_HandleTypeDef * huart uint32_t Polarity: { UART_DE_POLARITY_HIGH, UART_DE_POLARITY_LOW } uint32_t AssertionTime uint32_t DeassertionTime
输出	无
返回值	HAL_StatusTypeDef
资源使用	

说明	
----	--

3.24.42 HAL_UARTEx_WakeupCallback

3.24.42.1 功能介绍

UART 从 STOP 模式唤醒回调。

3.24.42.2 接口定义

函数接口	void HAL_UARTEx_WakeupCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.43 HAL_UARTEx_RxFifoFullCallback

3.24.43.1 功能介绍

UART 接收 FIFO 满回调。

3.24.43.2 接口定义

函数接口	void HAL_UARTEx_RxFifoFullCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.44 HAL_UARTEx_TxFifoEmptyCallback

3.24.44.1 功能介绍

UART 发送 FIFO 空回调。

3.24.44.2 接口定义

函数接口	void HAL_UARTEx_TxFifoEmptyCallback (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	无
资源使用	
说明	

3.24.45 HAL_UARTEx_RxEventCallback

3.24.45.1 功能介绍

UART 接收事件回调。

3.24.45.2 接口定义

函数接口	void HAL_UARTEx_RxEventCallback (UART_HandleTypeDef * huart,
------	---

	uint16_t Size)
输入	UART_HandleTypeDef * huart uint16_t Size
输出	无
返回值	无
资源使用	
说明	

3.24.46 HAL_UARTEx_StopModeWakeUpSourceConfig

3.24.46.1 功能介绍

设置从停止模式的唤醒中断标志选择。

3.24.46.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_StopModeWakeUpSourceConfig (UART_HandleTypeDef * huart, UART_WakeUpTypeDef WakeUpSelection)
输入	UART_HandleTypeDef * huart UART_WakeUpTypeDef WakeUpSelection: { UART_WAKEUP_ON_ADDRESS, UART_WAKEUP_ON_STARTBIT, UART_WAKEUP_ON_READDATA_NONEMPTY }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.47 HAL_UARTEx_EnableStopMode

3.24.47.1 功能介绍

使能 UART 停止模式。

3.24.47.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_EnableStopMode (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.48 HAL_UARTEx_DisableStopMode

3.24.48.1 功能介绍

禁止使能 UART 停止模式。

3.24.48.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_DisableStopMode (UART_HandleTypeDef * huart)
------	--

输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.49 HAL_MultiProcessorEx_AddressLength_Set

3.24.49.1 功能介绍

默认情况下，在多处理器模式下，当唤醒方法设置为地址标记时，UART 只处理 4 位长地址检测;这个 API 允许更长的地址检测。

3.24.49.2 接口定义

函数接口	HAL_StatusTypeDef HAL_MultiProcessorEx_AddressLength_Set (UART_HandleTypeDef * huart, uint32_t AddressLength)
输入	UART_HandleTypeDef * huart uint32_t AddressLength: { UART_ADDRESS_DETECT_4B, UART_ADDRESS_DETECT_7B }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.50 HAL_UARTEx_EnableFifoMode

3.24.50.1 功能介绍

使能 UART FIFO 模式。

3.24.50.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_EnableFifoMode (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.51 HAL_UARTEx_DisableFifoMode

3.24.51.1 功能介绍

禁止使能 UART FIFO 模式。

3.24.51.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_DisableFifoMode (UART_HandleTypeDef * huart)
输入	UART_HandleTypeDef * huart
输出	无
返回值	HAL_StatusTypeDef

资源使用	
说明	

3.24.52 HAL_UARTEx_SetTxFifoThreshold

3.24.52.1 功能介绍

设置 UART 发送 FIFO 阈值。

3.24.52.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_SetTxFifoThreshold (UART_HandleTypeDef * huart, uint32_t Threshold)
输入	UART_HandleTypeDef * huart uint32_t Threshold: { UART_FIFOTHRESHOLD_1_8, UART_FIFOTHRESHOLD_1_4, UART_FIFOTHRESHOLD_1_2, UART_FIFOTHRESHOLD_3_4, UART_FIFOTHRESHOLD_7_8, UART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.53 HAL_UARTEx_SetRxFifoThreshold

3.24.53.1 功能介绍

设置 UART 接收 FIFO 阈值。

3.24.53.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_SetRxFifoThreshold (UART_HandleTypeDef * huart, uint32_t Threshold)
输入	UART_HandleTypeDef * huart uint32_t Threshold: { UART_FIFOTHRESHOLD_1_8, UART_FIFOTHRESHOLD_1_4, UART_FIFOTHRESHOLD_1_2, UART_FIFOTHRESHOLD_3_4, UART_FIFOTHRESHOLD_7_8, UART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.54 HAL_UARTEx_ReceiveToIdle

3.24.54.1 功能介绍

以阻塞模式接收一定量的数据，直到接收到预期数量的数据或发生 IDLE 事件。

3.24.54.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_ReceiveToIdle (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size, uint16_t * RxLen, uint32_t Timeout)
输入	UART_HandleTypeDef * huart

	uint8_t * pData uint16_t Size uint16_t * RxLen uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.55 HAL_UARTEx_ReceiveToIdle_IT

3.24.55.1 功能介绍

在中断模式中接收一定数量的数据，直到接收到预期的数据数量或发生 IDLE 事件。

3.24.55.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_ReceiveToIdle_IT (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.24.56 HAL_UARTEx_ReceiveToIdle_DMA

3.24.56.1 功能介绍

在 DMA 模式中接收一定数量的数据，直到接收到预期的数据数量或发生 IDLE 事件。

3.24.56.2 接口定义

函数接口	HAL_StatusTypeDef HAL_UARTEx_ReceiveToIdle_DMA (UART_HandleTypeDef * huart, uint8_t * pData, uint16_t Size)
输入	UART_HandleTypeDef * huart uint8_t * pData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25 USART 模块

宏定义	数值	含义
HAL_USART_StateTypeDef		
HAL_USART_STATE_RESET	0x00	复位

HAL_USART_STATE_READY	0x01	就绪
HAL_USART_STATE_BUSY	0x02	忙
HAL_USART_STATE_BUSY_TX	0x12	发送忙
HAL_USART_STATE_BUSY_RX	0x22	接收忙
HAL_USART_STATE_BUSY_TX_RX	0x32	发送和接收忙
HAL_USART_STATE_TIMEOUT	0x03	错误
HAL_USART_STATE_ERROR	0x04	中止

属性	类型	字段名	含义
USART_TypeDef			
读写	uint32_t	CR1	控制寄存器 1
读写	uint32_t	CR2	控制寄存器 2
读写	uint32_t	CR3	控制寄存器 3
读写	uint32_t	BRR	波特率分频寄存器
读写	uint32_t	GTPR	保护时间和预分频器寄存器
读写	uint32_t	RTOR	超时及块传输长度寄存器
只写	uint32_t	RQR	请求寄存器
只读	uint32_t	ISR	中断和状态寄存器
只写	uint32_t	ICR	中断标志清零

			寄存器
只读	uint32_t	RDR	接收数据寄存器
读写	uint32_t	TDR	发送数据寄存器
读写	uint32_t	PRESC	预分频器寄存器
USART_InitTypeDef			
读写	uint32_t	BaudRate	波特率
读写	uint32_t	WordLength	字符长度
读写	uint32_t	StopBits	停止位
读写	uint32_t	Parity	RS485收发器使能信号极性
读写	uint32_t	Mode	模式
读写	uint32_t	CLKPolarity	时钟极性
读写	uint32_t	CLKPhase	时钟相位
读写	uint32_t	CLKLastBit	时钟传输最后一位
读写	uint32_t	ClockPrescaler	时钟分频值
USART_HandleTypeDef			
读写	USART_TypeDef	Instance	寄存器地址

读 写	UARTUSART_InitTypeDefInitTypeDef	Init	初始化参数
读 写	uint8_t	pTxBuffPtr	TX 传输缓冲区的指针
读 写	uint16_t	TxXferSize	传输发送大小
读 写	uint16_t	TxXferCount	传输发送计数器
读 写	uint16_t	pRxBuffPtr	RX 传输缓冲区的指针
读 写	uint16_t	RxXferSize	传输接收大小
读 写	uint16_t	RxXferCount	传输接收计数器
读 写	uint16_t	Mask	USART Rx RDR 注册掩码
读 写	uint16_t	NbRxDataToProcess	在 RX ISR 执行期间要处理的数据数
读 写	uint16_t	NbTxDataToProcess	在 TX ISR 执行期间要处理的数据

			数
读 写	uint32_t	SlaveMode	从模式
读 写	uint32_t	FifoMode	FIFO 模式
读 写	HAL_UART_RxTypeTypeDef	ReceptionType	持续接 收类型
读 写	uint16_t	NbRxDataToProcess	在 RX ISR 执 行期间 要处理 的数据 数
读 写	uint16_t	NbTxDataToProcess	在 TX ISR 执 行期间 要处理 的数据 数
读 写	指针函数	void(*RxISR)(struct __USART_HandleTypeDef *husart)	Rx ISR 函数指 针
读 写	指针函数	void(*TxISR)(struct __USART_HandleTypeDef *husart)	Tx ISR 函数指 针
读 写	DMA_HandleTypeDef	hdmatx	DMA 句柄
读 写	DMA_HandleTypeDef	hdmarx	DMA 句柄
读 写	HAL_LockTypeDef	Lock	锁定
读 写	HAL_UART_StateTypeDef	gState	USART 状态
读 写	uint32_t	ErrorCode	错误代 码

读 写	指针函数	void(*TxHalfCpltCallback)(struct __USART_HandleTypeDef *husart)	发送传 输完成 一半回 调
读 写	指针函数	void(*TxCpltCallback)(struct __USART_HandleTypeDef *husart)	发送传 输完成 回调
读 写	指针函数	void(*RxHalfCpltCallback)(struct __USART_HandleTypeDef *husart)	接收传 输完成 一半回 调
读 写	指针函数	void(*RxCpltCallback)(struct __USART_HandleTypeDef *husart)	接收传 输完成 回调
读 写	指针函数	void(*TxRxCpltCallback)(struct __USART_HandleTypeDef *husart)	发送接 收传输 完成回 调
读 写	指针函数	void(*ErrorCallback)(struct __USART_HandleTypeDef *husart)	错误回 调
读 写	指针函数	void(*AbortCpltCallback)(struct __USART_HandleTypeDef *husart)	中止完 成回调
读 写	指针函数	void(*RxFifoFullCallback)(struct __UART_HandleTypeDef *huart)	接 收 FIFO 满回调
读 写	指针函数	void(*TxFifoEmptyCallback)(struct __UART_HandleTypeDef *huart)	发 送 FIFO 空回调
读 写	指针函数	voi(*MspInitCallback)(struct __UART_HandleTypeDef *huart)	MSP 初 始化回 调
读 写	指针函数	void(*MspDeInitCallback)(struct __UART_HandleTypeDef *huart)	MSP 非 初始化 回调

3.25.1 HAL_USART_Init

3.25.1.1 功能介绍

初始化 USART。

3.25.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Init (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.2 HAL_USART_DeInit

3.25.2.1 功能介绍

清除 USART 初始化参数。

3.25.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_DeInit (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.3 HAL_USART_MspInit

3.25.3.1 功能介绍

USART MSP 初始化。

3.25.3.2 接口定义

函数接口	void HAL_USART_MspInit (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.4 HAL_USART_MspDeInit

3.25.4.1 功能介绍

清除 USART MSP 初始化参数。

3.25.4.2 接口定义

函数接口	void HAL_USART_MspInit (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart

输出	无
返回值	无
资源使用	
说明	

3.25.5 HAL_USART_Transmit

3.25.5.1 功能介绍

USART 在阻塞方式发送数据。

3.25.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Transmit (USART_HandleTypeDef * husart, uint8_t * pTXData, uint16_t Size, uint32_t Timeout)
输入	USART_HandleTypeDef * husart uint8_t * pTXData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.6 HAL_USART_Receive

3.25.6.1 功能介绍

USART 在阻塞方式接收数据。

3.25.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Receive (USART_HandleTypeDef * husart, uint8_t * pRXData, uint16_t Size, uint32_t Timeout)
输入	USART_HandleTypeDef * husart uint8_t * pRXData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.7 HAL_USART_TransmitReceive

3.25.7.1 功能介绍

USART 在阻塞方式发送和接收数据。

3.25.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_TransmitReceive (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size, uint32_t Timeout)
------	---

输入	USART_HandleTypeDef * husart uint8_t * pTxData uint8_t * pRXData uint16_t Size uint32_t Timeout
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.8 HAL_USART_Transmit_IT

3.25.8.1 功能介绍

USART 在阻塞方式以中断模式发送数据。

3.25.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Transmit_IT (USART_HandleTypeDef * husart, uint8_t * pTXData, uint16_t Size)
输入	USART_HandleTypeDef * husart uint8_t * pTXData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.9 HAL_USART_Receive_IT

3.25.9.1 功能介绍

USART 在阻塞方式以中断模式接收数据。

3.25.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Receive_IT (USART_HandleTypeDef * husart, uint8_t * pRXData, uint16_t Size)
输入	USART_HandleTypeDef * husart uint8_t * pRXData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.10 HAL_USART_TransmitReceive_IT

3.25.10.1 功能介绍

USART 在阻塞方式以中断模式发送和接收数据。

3.25.10.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_TransmitReceive_IT (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
输入	USART_HandleTypeDef * husart uint8_t * pTxData uint8_t * pRXData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.11 HAL_USART_Transmit_DMA

3.25.11.1 功能介绍

USART 在阻塞方式以 DMA 模式发送数据。

3.25.11.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Transmit_DMA (USART_HandleTypeDef * husart, uint8_t * pTXData, uint16_t Size)
输入	USART_HandleTypeDef * husart uint8_t * pTXData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.12 HAL_USART_Receive_DMA

3.25.12.1 功能介绍

USART 在阻塞方式以 DMA 模式接收数据。

3.25.12.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Receive_DMA (USART_HandleTypeDef * husart, uint8_t * pRXData, uint16_t Size)
输入	USART_HandleTypeDef * husart uint8_t * pRXData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.13 HAL_USART_TransmitReceive_DMA

3.25.13.1 功能介绍

USART 在阻塞方式以 DMA 模式发送和接收数据。

3.25.13.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_TransmitReceive_DMA (USART_HandleTypeDef * husart, uint8_t * pTxData, uint8_t * pRxData, uint16_t Size)
输入	USART_HandleTypeDef * husart uint8_t * pTxData uint8_t * pRXData uint16_t Size
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.14 HAL_USART_DMAPause

3.25.14.1 功能介绍

暂停 DMA 传输。

3.25.14.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_DMAPause (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.15 HAL_USART_DMAResume

3.25.15.1 功能介绍

恢复 DMA 传输。

3.25.15.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_DMAResume (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.16 HAL_USART_DMAStop

3.25.16.1 功能介绍

停止 DMA 传输。

3.25.16.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_DMAStop (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.17 HAL_USART_Abort

3.25.17.1 功能介绍

中止正在进行的传输(阻塞模式)。

3.25.17.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Abort (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.18 HAL_USART_Abort_IT

3.25.18.1 功能介绍

以中断模式中中止正在进行的传输(阻塞模式)。

3.25.18.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USART_Abort_IT (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.19 HAL_USART_IRQHandler

3.25.19.1 功能介绍

处理 USART 的中断请求。

3.25.19.2 接口定义

函数接口	void HAL_USART_IRQHandler (USART_HandleTypeDef * husart)
------	---

输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.20 HAL_USART_TxHalfCpltCallback

3.25.20.1 功能介绍

USART 发送完成一半回调。

3.25.20.2 接口定义

函数接口	void HAL_USART_TxHalfCpltCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.21 HAL_USART_TxCpltCallback

3.25.21.1 功能介绍

USART 发送完成回调。

3.25.21.2 接口定义

函数接口	void HAL_USART_TxCpltCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.22 HAL_USART_RxHalfCpltCallback

3.25.22.1 功能介绍

USART 接收完成一半回调。

3.25.22.2 接口定义

函数接口	void HAL_USART_RxHalfCpltCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.23 HAL_USART_RxCpltCallback

3.25.23.1 功能介绍

USART 接收完成回调。

3.25.23.2 接口定义

函数接口	void HAL_USART_RxCpltCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.24 HAL_USART_TxRxCpltCallback

3.25.24.1 功能介绍

USART 发送接收完成回调。

3.25.24.2 接口定义

函数接口	void HAL_USART_TxRxCpltCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.25 HAL_USART_ErrorCallback

3.25.25.1 功能介绍

USART 错误回调。

3.25.25.2 接口定义

函数接口	void HAL_USART_ErrorCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.26 HAL_USART_AbortCpltCallback

3.25.26.1 功能介绍

USART 中止完成回调。

3.25.26.2 接口定义

函数接口	void HAL_USART_AbortCpltCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart

输出	无
返回值	无
资源使用	
说明	

3.25.27 HAL_USART_GetState

3.25.27.1 功能介绍

获取 USART 状态。

3.25.27.2 接口定义

函数接口	HAL_USART_StateTypeDef HAL_USART_GetState (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_USART_StateTypeDef
资源使用	
说明	

3.25.28 HAL_USART_GetError

3.25.28.1 功能介绍

获取 USART 错误。

3.25.28.2 接口定义

函数接口	uint32_t HAL_USART_GetError (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	错误值
资源使用	
说明	

3.25.29 HAL_USARTEEx_RxFifoFullCallback

3.25.29.1 功能介绍

USART 接收 FIFO 满回调。

3.25.29.2 接口定义

函数接口	void HAL_USARTEEx_RxFifoFullCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.30 HAL_USARTEEx_TxFifoEmptyCallback

3.25.30.1 功能介绍

USART 发送 FIFO 空回调。

3.25.30.2 接口定义

函数接口	void HAL_USARTEEx_TxFifoEmptyCallback (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	无
资源使用	
说明	

3.25.31 HAL_USARTEEx_EnableSlaveMode

3.25.31.1 功能介绍

使能 SPI 从模式。

3.25.31.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_EnableSlaveMode (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.32 HAL_USARTEEx_DisableSlaveMode

3.25.32.1 功能介绍

禁止使能 SPI 从模式。

3.25.32.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_DisableSlaveMode (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.33 HAL_USARTEEx_ConfigNSS

3.25.33.1 功能介绍

配置从选择输入引脚(NSS)。

3.25.33.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_ConfigNSS
------	--

	(USART_HandleTypeDef * husart, uint32_t NSSConfig)
输入	USART_HandleTypeDef * husart uint32_t NSSConfig: { USART_NSS_HARD, USART_NSS_SOFT }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.34 HAL_USARTEEx_EnableFifoMode

3.25.34.1 功能介绍

使能 USART FIFO 模式。

3.25.34.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_EnableFifoMode (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.35 HAL_USARTEEx_DisableFifoMode

3.25.35.1 功能介绍

禁止使能 USART FIFO 模式。

3.25.35.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_DisableFifoMode (USART_HandleTypeDef * husart)
输入	USART_HandleTypeDef * husart
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.36 HAL_USARTEEx_SetTxFifoThreshold

3.25.36.1 功能介绍

设置 USART 发送 FIFO 阈值。

3.25.36.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_SetTxFifoThreshold (USART_HandleTypeDef * husart, uint32_t Threshold)
输入	USART_HandleTypeDef * husart uint32_t Threshold: { USART_FIFOTHRESHOLD_1_8, USART_FIFOTHRESHOLD_1_4,

	USART_FIFOTHRESHOLD_1_2, USART_FIFOTHRESHOLD_3_4, USART_FIFOTHRESHOLD_7_8, USART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.25.37 HAL_USARTEEx_SetRxFifoThreshold

3.25.37.1 功能介绍

设置 USART 接收 FIFO 阈值。

3.25.37.2 接口定义

函数接口	HAL_StatusTypeDef HAL_USARTEEx_SetRxFifoThreshold (USART_HandleTypeDef * husart, uint32_t Threshold)
输入	USART_HandleTypeDef * husart uint32_t Threshold: { USART_FIFOTHRESHOLD_1_8, USART_FIFOTHRESHOLD_1_4, USART_FIFOTHRESHOLD_1_2, USART_FIFOTHRESHOLD_3_4, USART_FIFOTHRESHOLD_7_8, USART_FIFOTHRESHOLD_8_8 }
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.26 WWDG 模块

属性	类型	字段名	含义
WWDG_TypeDef			
读写	uint32_t	CR	控制寄存器
读写	uint32_t	CFR	配置寄存器
读写	uint32_t	SR	状态寄存器
WWDG_InitTypeDef			
读写	uint32_t	Prescaler	分频值
读写	uint32_t	Window	窗口值
读写	uint32_t	Counter	计数值
读写	uint32_t	EWIMode	提前唤醒模式
WWDG_HandleTypeDef			
读写	WWDG_TypeDef	Instance	寄存器地址
读写	WWDG_InitTypeDef	Init	初始化参数
读写	指针函数	void(*EwiCallback)(struct __WWDG_HandleTypeDef	提前唤醒回调

		*hwwdg)	
读写	指针函数	void(*MspInitCallback)(struct __WWDG_HandleTypeDef *hwwdg)	MSP 初始化回调

3.26.1 HAL_WWDG_Init

3.26.1.1 功能介绍

初始化 WWDG。

3.26.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_WWDG_Init (WWDG_HandleTypeDef * hwwdg)
输入	WWDG_HandleTypeDef * hwwdg
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.26.2 HAL_WWDG_MspInit

3.26.2.1 功能介绍

初始化 WWDG MSP。

3.26.2.2 接口定义

函数接口	void HAL_WWDG_MspInit (WWDG_HandleTypeDef * hwwdg)
输入	WWDG_HandleTypeDef * hwwdg
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.26.3 HAL_WWDG_Refresh

3.26.3.1 功能介绍

刷新 WWDG。

3.26.3.2 接口定义

函数接口	HAL_StatusTypeDef HAL_WWDG_Refresh (WWDG_HandleTypeDef * hwwdg)
输入	WWDG_HandleTypeDef * hwwdg
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.26.4 HAL_WWDG_IRQHandler

3.26.4.1 功能介绍

处理 WWDG 中断请求。

3.26.4.2 接口定义

函数接口	void HAL_WWDG_IRQHandler (WWDG_HandleTypeDef * hwwdg)
输入	WWDG_HandleTypeDef * hwwdg
输出	无
返回值	无
资源使用	
说明	

3.26.5 HAL_WWDG_EarlyWakeupCallback

3.26.5.1 功能介绍

WWDG 提前唤醒回调。

3.26.5.2 接口定义

函数接口	void HAL_WWDG_EarlyWakeupCallback (WWDG_HandleTypeDef * hwwdg)
输入	WWDG_HandleTypeDef * hwwdg
输出	无
返回值	无
资源使用	
说明	

3.27 CORDIC 模块

属性	类型	字段名	含义
CORDIC_InitTypeDef			
读写	uint32_t	Function	功能函数
读写	uint32_t	InSize	输入数据长度
读写	uint32_t	OutSize	输出数据长度
读写	uint32_t	NbWrite	参数数量
读写	uint32_t	NbRead	结果数读取次数
读写	uint32_t	Precision	迭代次数
CORDIC_HandleTypeDef			
读写	CORDIC_TypeDef	Instance	寄存器地址
读写	int32_t*	pInBuff	CORDIC 输

			入数据地址
读写	int32_t *	pOutBuff	CORDIC 输出数据地址
读写	uint32_t	NbCalcToOrder	计算数排列
读写	uint32_t	NbCalcToGet	获取的计算结果
读写	uint32_t	DMADirection	CORDIC DMA 传输方向
读写	DMA_HandleTypeDef *	hdmaIn	通过 DMA 的数据输入句柄
读写	DMA_HandleTypeDef *	hdmaOut	通过 DMA 的数据输出句柄
读写	HAL_LockTypeDef	Lock	CORDIC 对象锁
读写	HAL_CORDIC_StateTypeDef	State	CORDIC 状态
读写	uint32_t	ErrorCode	CORDIC 错误码
读写	指针函数	void (* ErrorCallback)(struct __CORDIC_HandleTypeDef *hcordic)	错误回调
读写	指针函数	void (* CalculateCpltCallback)(struct __CORDIC_HandleTypeDef *hcordic)	计算完成回调
读写	指针函数	void (* MspInitCallback)(struct __CORDIC_HandleTypeDef *hcordic)	MSP 初始化回调
读写	指针函数	void (* MspDeInitCallback)(struct __CORDIC_HandleTypeDef *hcordic)	MSP 析构回调

3.27.1 HAL_CORDIC_Init

3.27.1.1 功能介绍

初始化 CORDIC。

3.27.1.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_Init(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.2 HAL_CORDIC_DeInit

3.27.2.1 功能介绍

析构化 CORDIC。

3.27.2.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_DeInit(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.3 HAL_CORDIC_MspInit

3.27.3.1 功能介绍

初始化 CORDIC MSP。

3.27.3.2 接口定义

函数接口	void HAL_CORDIC_MspInit(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	无
资源使用	
说明	

3.27.4 HAL_CORDIC_MspDeInit

3.27.4.1 功能介绍

析构化 CORDIC MSP。

3.27.4.2 接口定义

函数接口	void HAL_CORDIC_MspDeInit(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC

输出	无
返回值	无
资源使用	
说明	

3.27.5 HAL_CORDIC_Configure

3.27.5.1 功能介绍

CORDIC 配置。

3.27.5.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_Configure(CORDIC_HandleTypeDef *hcordic, CORDIC_ConfigTypeDef *sConfig)
输入	CORDIC_HandleTypeDef *hcordic CORDIC_ConfigTypeDef *sConfig
输出	无
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.6 HAL_CORDIC_Calculate

3.27.6.1 功能介绍

轮询方式计算 CORDIC。

3.27.6.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_Calculate(CORDIC_HandleTypeDef *hcordic, int32_t *pInBuff, int32_t *pOutBuff, uint32_t NbCalc, uint32_t Timeout);
输入	CORDIC_HandleTypeDef *hcordic int32_t *pInBuff: uint32_t NbCalc: [0,0xffffffff] uint32_t Timeout: [0,0xffffffff]
输出	int32_t *pOutBuff:
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.7 HAL_CORDIC_CalculateZO

3.27.7.1 功能介绍

轮询 Zero-overhead 方式计算 CORDIC。

3.27.7.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_CalculateZO(CORDIC_HandleTypeDef *hcordic, int32_t *pInBuff, int32_t *pOutBuff, uint32_t NbCalc, uint32_t Timeout)
------	--

输入	CORDIC_HandleTypeDef *hcordic int32_t *pInBuff: uint32_t NbCalc: [0,0xffffffff] uint32_t Timeout: [0,0xffffffff]
输出	int32_t *pOutBuff:
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.8 HAL_CORDIC_Calculate_IT

3.27.8.1 功能介绍

中断方式计算 CORDIC。

3.27.8.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_Calculate_IT(CORDIC_HandleTypeDef *hcordic, int32_t *pInBuff, int32_t *pOutBuff, uint32_t NbCalc)
输入	CORDIC_HandleTypeDef *hcordic int32_t *pInBuff: uint32_t NbCalc: [0,0xffffffff]
输出	int32_t *pOutBuff:
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.9 HAL_CORDIC_Calculate_DMA

3.27.9.1 功能介绍

DMA 方式计算 CORDIC。

3.27.9.2 接口定义

函数接口	HAL_StatusTypeDef HAL_CORDIC_Calculate_DMA(CORDIC_HandleTypeDef *hcordic, int32_t *pInBuff, int32_t *pOutBuff, uint32_t NbCalc, uint32_t DMADirection)
输入	CORDIC_HandleTypeDef *hcordic int32_t *pInBuff: uint32_t NbCalc: [0,0xffffffff] uint32_t DMADirection: { CORDIC_DMA_DIR_NONE , CORDIC_DMA_DIR_IN , CORDIC_DMA_DIR_OUT , CORDIC_DMA_DIR_IN_OUT }
输出	int32_t *pOutBuff:
返回值	HAL_StatusTypeDef
资源使用	
说明	

3.27.10 HAL_CORDIC_ErrorCallback

3.27.10.1 功能介绍

CORDIC 计算出错回调。

3.27.10.2 接口定义

函数接口	void HAL_CORDIC_ErrorCallback(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	无
资源使用	
说明	

3.27.11 HAL_CORDIC_CalculateCpltCallback

3.27.11.1 功能介绍

CORDIC 计算完成回调。

3.27.11.2 接口定义

函数接口	void HAL_CORDIC_CalculateCpltCallback(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	无
资源使用	
说明	

3.27.12 HAL_CORDIC_IRQHandler

3.27.12.1 功能介绍

处理 CORDIC 的中断请求。

3.27.12.2 接口定义

函数接口	void HAL_CORDIC_IRQHandler(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	无
资源使用	
说明	

3.27.13 HAL_CORDIC_GetState

3.27.13.1 功能介绍

获取 CORDIC 状态。

3.27.13.2 接口定义

函数接口	HAL_CORDIC_StateTypeDef HAL_CORDIC_GetState(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC

输出	无
返回值	HAL_CORDIC_StateTypeDef
资源使用	
说明	

3.27.14 HAL_CORDIC_GetError

3.27.14.1 功能介绍

获取 CORDIC 错误。

3.27.14.2 接口定义

函数接口	uint32_t HAL_CORDIC_GetError(CORDIC_HandleTypeDef *hCORDIC)
输入	CORDIC_HandleTypeDef *hCORDIC
输出	无
返回值	错误值
资源使用	
说明	