

32 位微控制器

KF32A 标准固件库 API 使用手册

目 录

目 录.....	1
图示列表.....	26
表格列表.....	27
1 文档说明.....	50
1.1 简介.....	50
1 外设缩写说明.....	50
1.2 命名规则.....	51
1.3 编码规则.....	51
1.3.1 枚举型.....	51
1.3.2 结构体.....	51
1.3.3 宏定义.....	52
1.4 固件库描述.....	53
2 模数转换模块（ADC）.....	54
2.1 文件引用关系.....	54
2.2 ADC 寄存器结构.....	54
2.3 ADC 宏定义.....	57
2.4 ADC 库函数.....	57
2.4.1 函数 ADC_Reset.....	59
2.4.2 函数 ADC_Configuration.....	59
2.4.3 函数 ADC_Struct_Init.....	60
2.4.4 函数 ADC_Delay_Configuration.....	60
2.4.5 函数 ADC_Delay_Struct_Init.....	60
2.4.6 函数 ADC_Cmd.....	61
2.4.7 函数 ADC_Analog_Watchdog_Configuration.....	61
2.4.8 函数 ADC_Analog_Watchdog_Struct_Init.....	61
2.4.9 函数 ADC_Watchdog_Single_Channel_Enable.....	61
2.4.10 函数 ADC_Scan_Mode_Enable.....	62
2.4.11 函数 ADC_Bossel_Comparator_Calibration.....	62
2.4.12 函数 ADC_Bossel_Calibration.....	62
2.4.13 函数 ADC_Trim_Current_Intensity_Bias.....	63
2.4.14 函数 ADC_Analog_Clock_Config.....	63
2.4.15 函数 ADC_Data_Align_Config.....	63
2.4.16 函数 ADC_Clock_Source_Config.....	64
2.4.17 函数 ADC-Regular_Channel_DMA_Cmd.....	64
2.4.18 函数 ADC_High_Priority_Channel_DMA_Cmd.....	65
2.4.19 函数 ADC_Double_Mode_Config.....	65
2.4.20 函数 ADC_Reference_Voltage_Config.....	66
2.4.21 函数 ADC_Analog_Watchdog_Channel_Config.....	66

2.4.22	函数 ADC_Regular_Channel_Watchdog_Enable.....	66
2.4.23	函数 ADC_External_Trig_Conv_Enable.....	67
2.4.24	函数 ADC_External_Trig_Conv_Config.....	67
2.4.25	函数 ADC_Regular_Channel_Config.....	69
2.4.26	函数 ADC_Regular-Sequencer_Length_Config.....	69
2.4.27	函数 ADC_Regular_Channel_TxCCRy_Trig_Enable.....	70
2.4.28	函数 ADC_Software_Start_Conv.....	70
2.4.29	函数 ADC_Continuous_Mode_Cmd.....	71
2.4.30	函数 ADC_Disc_Mode_Channel_Count_Config.....	71
2.4.31	函数 ADC_Disc_Mode_Cmd.....	71
2.4.32	函数 ADC_Get_Conversion_Value.....	72
2.4.33	函数 ADC_High_Priority_Watchdog_Enable.....	72
2.4.34	函数 ADC_HPExternal_Trig_Conv_Enable.....	72
2.4.35	函数 ADC_High_Priority_Channel_Config.....	72
2.4.36	函数 ADC_High_Priority-Sequencer_Length_Config.....	73
2.4.37	函数 ADC_Set_HPChannel_Conv_Value_Offset.....	73
2.4.38	函数 ADC_HPExternal_Trig_Conv_Config.....	74
2.4.39	函数 ADC_Software_HPStart_Conv.....	76
2.4.40	函数 ADC_HPAuto_Conv_Cmd.....	76
2.4.41	函数 ADC_HPDisc_Mode_Cmd.....	76
2.4.42	函数 ADC_Get_HPConversion_Data.....	76
2.4.43	函数 ADC_Set_INT_Enable.....	77
2.4.44	函数 ADC_Get_INT_Flag.....	77
2.4.45	函数 ADC_Clear_INT_Flag.....	78
2.4.46	函数 ADC_Get_INT_Status.....	78
3	高级定时器(ATIM).....	79
3.1	文件引用关系.....	79
3.2	ATIM/ECCP 寄存器结构.....	79
3.3	ATIM/ECCP 库函数.....	83
3.3.1	函数 ATIM_X_Configuration.....	87
3.3.2	函数 ATIM_Z_Configuration.....	87
3.3.3	函数 ATIM_Struct_Init.....	87
3.3.4	函数 ATIM_X_Cmd.....	88
3.3.5	函数 ATIM_Z_Cmd.....	88
3.3.6	函数 ATIM_X_Updata_Configuration.....	88
3.3.7	函数 ATIM_Z_Updata_Configuration.....	88
3.3.8	函数 ATIM_X_Updata_Cmd.....	89
3.3.9	函数 ATIM_Z_Updata_Cmd.....	89
3.3.10	函数 ATIM_X_Set_Counter.....	89
3.3.11	函数 ATIM_Z_Set_Counter.....	90
3.3.12	函数 ATIM_X_Set_Period.....	90
3.3.13	函数 ATIM_Z_Set_Period.....	90
3.3.14	函数 ATIM_X_Set_Prescaler.....	91

3.3.15	函数 ATIM_Z_Set_Prescaler.....	91
3.3.16	函数 ATIM_X_Counter_Mode_Config.....	91
3.3.17	函数 ATIM_Z_Counter_Mode_Config.....	92
3.3.18	函数 ATIM_X_Clock_Config.....	92
3.3.19	函数 ATIM_Z_Clock_Config.....	92
3.3.20	函数 ATIM_X_Postscaler_Config.....	93
3.3.21	函数 ATIM_Z_Postscaler_Config.....	93
3.3.22	函数 ATIM_X_External_Pulse_Sync_Config.....	94
3.3.23	函数 ATIM_Z_External_Pulse_Sync_Config.....	94
3.3.24	函数 ATIM_X_Work_Mode_Config.....	95
3.3.25	函数 ATIM_Z_Work_Mode_Config.....	95
3.3.26	函数 ATIM_X_Get_Direction.....	96
3.3.27	函数 ATIM_Z_Get_Direction.....	96
3.3.28	函数 ATIM_X_Overflow_AD_Enable.....	96
3.3.29	函数 ATIM_Z_Overflow_AD_Enable.....	96
3.3.30	函数 ATIM_X_Underflow_AD_Enable.....	97
3.3.31	函数 ATIM_Z_Underflow_AD_Enable.....	97
3.3.32	函数 ATIM_X_TriggerAD_Config.....	97
3.3.33	函数 ATIM_Z_TriggerAD_Config.....	98
3.3.34	函数 ATIM_X_Set_TriggerAD_Signal.....	98
3.3.35	函数 ATIM_Z_Set_TriggerAD_Signal.....	98
3.3.36	函数 ATIM_X_Update_Immediately_Config.....	99
3.3.37	函数 ATIM_Z_Update_Immediately_Config.....	99
3.3.38	函数 ATIM_X_Update_Output_Ctl.....	99
3.3.39	函数 ATIM_Z_Update_Output_Ctl.....	99
3.3.40	函数 ATIM_X_Update_Enable.....	100
3.3.41	函数 ATIM_Z_Update_Enable.....	100
3.3.42	函数 ATIM_X_Set_Update_Counter.....	100
3.3.43	函数 ATIM_Z_Set_Update_Counter.....	101
3.3.44	函数 ATIM_X_Slave_Mode_Config.....	101
3.3.45	函数 ATIM_Z_Slave_Mode_Config.....	101
3.3.46	函数 ATIM_Master_Mode_Config.....	102
3.3.47	函数 ATIM_Master_Slave_Snyc_Enable.....	102
3.3.48	函数 ATIM_Trigger_Select_Config.....	103
3.3.49	函数 ATIM_Timer_Unite_Enable.....	103
3.3.50	函数 ATIM_X_Get_Counter.....	103
3.3.51	函数 ATIM_Z_Get_Counter.....	104
3.3.52	函数 ATIM_X_Get_Period.....	104
3.3.53	函数 ATIM_Z_Get_Period.....	104
3.3.54	函数 ATIM_X_Get_Prescaler.....	105
3.3.55	函数 ATIM_Z_Get_Prescaler.....	105
3.3.56	函数 ATIM_X_Update_INT_Enable.....	105
3.3.57	函数 ATIM_Z_Update_INT_Enable.....	105
3.3.58	函数 ATIM_X_Overflow_INT_Enable.....	106

3.3.59	函数 ATIM_Z_Overflow_INT_Enable.....	106
3.3.60	函数 ATIM_X_Trigger_INT_Enable.....	106
3.3.61	函数 ATIM_X_Updata_DMA_Enable.....	107
3.3.62	函数 ATIM_Z_Updata_DMA_Enable.....	107
3.3.63	函数 ATIM_X_Trigger_DMA_Enable.....	107
3.3.64	函数 ATIM_X_Get_Updata_INT_Flag.....	108
3.3.65	函数 ATIM_Z_Get_Updata_INT_Flag.....	108
3.3.66	函数 ATIM_X_Get_Overflow_INT_Flag.....	108
3.3.67	函数 ATIM_Z_Get_Overflow_INT_Flag.....	108
3.3.68	函数 ATIM_X_Get_Trigger_INT_Flag.....	109
3.3.69	函数 ATIM_X_Generate_Trigger_Config.....	109
3.3.70	函数 ATIM_X_Get_Updata_DMA_INT_Flag.....	109
3.3.71	函数 ATIM_Z_Get_Updata_DMA_INT_Flag.....	110
3.3.72	函数 ATIM_X_Get_Trigger_DMA_INT_Flag.....	110
3.3.73	函数 ATIM_X_Clear_Updata_INT_Flag.....	110
3.3.74	函数 ATIM_Z_Clear_Updata_INT_Flag.....	110
3.3.75	函数 ATIM_X_Clear_Overflow_INT_Flag.....	111
3.3.76	函数 ATIM_Z_Clear_Overflow_INT_Flag.....	111
3.3.77	函数 ATIM_X_Clear_Trigger_INT_Flag.....	111
3.3.78	函数 ECCP_Compare_Configuration.....	111
3.3.79	函数 ECCP_Capture_Configuration.....	112
3.3.80	函数 ECCP_Capture_Struct_Init.....	112
3.3.81	函数 ECCP_PWM_Configuration.....	113
3.3.82	函数 ECCP_PWM_Struct_Init.....	113
3.3.83	函数 ECCP_Capture_Mode_Config.....	113
3.3.84	函数 ECCP_Compare_Mode_Config.....	114
3.3.85	函数 ECCP_PWM_Mode_Config.....	114
3.3.86	函数 ECCP_Get_Capture_Result.....	115
3.3.87	函数 ECCP_Get_Compare_Result.....	115
3.3.88	函数 ECCP_Set_Compare_Result.....	116
3.3.89	函数 ECCP_Generate_Trigger_Config.....	116
3.3.90	函数 ECCP_PWM_Input_Enable.....	117
3.3.91	函数 ECCP_Input_XOR_Enable.....	117
3.3.92	函数 ECCP_Single_Pulse_Enable.....	117
3.3.93	函数 ECCP_Single_Pulse_Shut_Enable.....	118
3.3.94	函数 ECCP_PWM_Restart_Enable.....	118
3.3.95	函数 ECCP_Dead_Time_Config.....	118
3.3.96	函数 ECCP_Channel_Output_Control.....	119
3.3.97	函数 ECCP_Channel_Output_Mode.....	119
3.3.98	函数 ECCP_Channel_Work_State_Config.....	120
3.3.99	函数 ECCP_Get_Channel_Work_State.....	120
3.3.100	函数 ECCP_CHANNEL4_Shutdown_SEL.....	121
3.3.101	函数 ECCP_Channel_Shutdown_Signal.....	121
3.3.102	函数 ECCP_Channel_Pin_Ctl.....	122

3.3.103	函数 ECCP_Zero_Clock_Config.....	123
3.3.104	函数 ECCP_Channel_Pin_Tristate_Enable.....	123
3.3.105	函数 ECCP_Channel_INT_Enable.....	124
3.3.106	函数 ECCP_X_Turn_off_DMA_Enable.....	124
3.3.107	函数 ECCP_Channel_DMA_Enable.....	125
3.3.108	函数 ECCP_Get_Channel_Trigger_INT_Flag.....	125
3.3.109	函数 ECCP_X_Get_Turn_off_DMA_Flag.....	125
3.3.110	函数 ECCP_Get_Trigger_DMA_INT_Flag.....	126
3.3.111	函数 ECCP_Clear_Channel_INT_Flag.....	126
3.3.112	函数 ECCP_PWM_Move_Phase_Enable.....	127
3.3.113	函数 ECCP_Channel_Zero_Detect_Sequential_Ctl.....	127
3.3.114	函数 ECCP_Get_Channel_Zero_Detection_State.....	127
3.3.115	函数 ECCP_Clear_Channel_Zero_Detection_State.....	128
3.3.116	函数 ECCP_Channel_Zero_Detect_Enable.....	128
3.3.117	函数 ECCP_Channel_Zero_Voltage_Config.....	129
4	备份域 (BKP)	130
4.1	文件引用关系.....	130
4.2	BKP 寄存器结构.....	130
4.3	BKP 宏定义.....	131
4.4	BKP 库函数.....	131
4.4.1	函数 BKP_Reset.....	131
4.4.2	函数 BKP_Write_And_Read_Enable.....	132
4.4.3	函数 BKP_Reset_Enable.....	132
4.4.4	函数 BKP_Pin_Effective_Level_Config.....	132
4.4.5	函数 BKP_Pin_Enable.....	133
4.4.6	函数 BKP_RTC_Clock_Config.....	133
4.4.7	函数 BKP_External_Clock_Bypass_Enable.....	134
4.4.8	函数 BKP_Data_Config.....	134
4.4.9	函数 BKP_Get_Data.....	134
4.4.10	函数 BKP_Pin_TAMP_INT_Enable.....	134
4.4.11	函数 BKP_Get_Pin_TAMP_INT_Flag.....	135
4.4.12	函数 BKP_Clear_Pin_TAMP_INT_Flag.....	135
5	基本定时器(BTIM).....	136
5.1	文件引用关系.....	136
5.2	BTIM 寄存器结构.....	136
5.3	BTIM 库函数.....	138
5.3.1	函数 TIM_Reset.....	139
5.3.2	函数 BTIM_Configuration.....	139
5.3.3	函数 BTIM_Struct_Init.....	139
5.3.4	函数 BTIM_Cmd.....	140
5.3.5	函数 BTIM_Set_Counter.....	140
5.3.6	函数 BTIM_Set_Period.....	140

5.3.7	函数 BTIM_Set_Prescaler.....	141
5.3.8	函数 BTIM_Counter_Mode_Config.....	141
5.3.9	函数 BTIM_Clock_Config.....	141
5.3.10	函数 BTIM_External_Pulse_Sync_Config.....	142
5.3.11	函数 BTIM_Work_Mode_Config.....	142
5.3.12	函数 BTIM_Generate_Trigger_Config.....	142
5.3.13	函数 BTIM_Single_Pulse_Enable.....	143
5.3.14	函数 BTIM_Single_Pulse_Shut_Enable.....	143
5.3.15	函数 BTIM_Update_Immediately_Config.....	143
5.3.16	函数 BTIM_Master_Slave_Sync_Config.....	144
5.3.17	函数 BTIM_Trigger_Select_Config.....	144
5.3.18	函数 BTIM_Slave_Mode_Config.....	144
5.3.19	函数 BTIM_Master_Mode_Config.....	145
5.3.20	函数 BTIM_Update_Rising_Edge_Config.....	145
5.3.21	函数 BTIM_Update_Enable.....	146
5.3.22	函数 BTIM_Get_Direction.....	146
5.3.23	函数 BTIM_Get_Counter.....	146
5.3.24	函数 BTIM_Get_Period.....	147
5.3.25	函数 BTIM_Get_Prescaler.....	147
5.3.26	函数 BTIM_Trigger_DMA_Enable.....	147
5.3.27	函数 BTIM_Update_DMA_Enable.....	147
5.3.28	函数 BTIM_Overflow_INT_Enable.....	148
5.3.29	函数 BTIM_Trigger_INT_Enable.....	148
5.3.30	函数 BTIM_Update_INT_Enable.....	148
5.3.31	函数 BTIM_Get_Trigger_DMA_INT_Status.....	149
5.3.32	函数 BTIM_Get_Update_DMA_INT_Status.....	149
5.3.33	函数 BTIM_Get_Overflow_INT_Status.....	149
5.3.34	函数 BTIM_Get_Trigger_INT_Status.....	149
5.3.35	函数 BTIM_Get_Update_INT_Status.....	150
5.3.36	函数 BTIM_Get_Trigger_DMA_INT_Flag.....	150
5.3.37	函数 BTIM_Get_Update_DMA_INT_Flag.....	150
5.3.38	函数 BTIM_Get_Overflow_INT_Flag.....	150
5.3.39	函数 BTIM_Get_Update_INT_Flag.....	151
5.3.40	函数 BTIM_Clear_Overflow_INT_Flag.....	151
5.3.41	函数 BTIM_Clear_Trigger_INT_Flag.....	151
5.3.42	函数 BTIM_Clear_Update_INT_Flag.....	151
6	控制器局域网 (CAN)	152
6.1	文件引用关系.....	152
6.2	CAN 寄存器结构体.....	152
6.3	CAN 宏定义.....	154
6.4	CAN 库函数.....	154
6.4.1	函数 CAN_Reset.....	155
6.4.2	函数 CAN_Configuration.....	155

6.4.3	函数 CAN_Struct_Init.....	156
6.4.4	函数 CAN_Get_Receive_Message_Counter.....	156
6.4.5	函数 CAN_Get_Transmit_Status.....	156
6.4.6	函数 CAN_Cmd.....	156
6.4.7	函数 CAN_Clock_Source_Config.....	157
6.4.8	函数 CAN_Sleep_Mode_Enable.....	157
6.4.9	函数 CAN_Reset_Mode_Enable.....	157
6.4.10	函数 CAN_Work_Mode_Config.....	158
6.4.11	函数 CAN_Bus_Sample_Times_Config.....	158
6.4.12	函数 CAN_Time_Segment_Config.....	158
6.4.13	函数 CAN_Sync_Jump_Width_Config.....	159
6.4.14	函数 CAN_Baud_Rate_Preset_Config.....	159
6.4.15	函数 CAN_Get_Error_Code.....	159
6.4.16	函数 CAN_Get_Error_Warning_Limit.....	160
6.4.17	函数 CAN_Get_Error_Counter.....	160
6.4.18	函数 CAN_Error_Warning_Limit_Config.....	160
6.4.19	函数 CAN_Error_Counter_Config.....	161
6.4.20	函数 CAN_Acceptance_Config.....	161
6.4.21	函数 CAN_Get_Acceptance.....	161
6.4.22	函数 CAN_Acceptance_Mask_Config.....	161
6.4.23	函数 CAN_Get_Acceptance_Mask.....	162
6.4.24	函数 CAN_Transmit_Message_Configuration.....	162
6.4.25	函数 CAN_Receive_Message_Configuration.....	162
6.4.26	函数 CAN_Message_Struct_Init.....	163
6.4.27	函数 CAN_Clear_Buffer_Overflow_Flag.....	163
6.4.28	函数 CAN_Release_Receive_Buffer.....	163
6.4.29	函数 CAN_Transmit_Single.....	163
6.4.30	函数 CAN_Transmit_Repeat.....	164
6.4.31	函数 CAN_Frame_Format_Config.....	164
6.4.32	函数 CAN_Remote_Request_Config.....	164
6.4.33	函数 CAN_Data_Length_Config.....	165
6.4.34	函数 CAN_Identification_Code_Config.....	165
6.4.35	函数 CAN_Get_INT_Flag.....	165
6.4.36	函数 CAN_Clear_INT_Flag.....	166
6.4.37	函数 CAN_Set_INT_Enable.....	166
7	模拟比较器模块(CMP).....	168
7.1	文件引用关系.....	168
7.2	CMP 寄存器结构.....	168
7.3	CMP 库函数.....	170
7.3.1	函数 CMP_Reset.....	171
7.3.2	函数 CMP_Configuration.....	171
7.3.3	函数 CMP_Struct_Init.....	171
7.3.4	函数 CMP0_Cmd.....	172

7.3.5	函数 CMP1_Cmd.....	172
7.3.6	函数 CMP2_Cmd.....	172
7.3.7	函数 CMP3_Cmd.....	172
7.3.8	函数 CMP0_POSITIVE_INPUT_SELECT.....	173
7.3.9	函数 CMP0_NEGATIVE_INPUT_SELECT.....	173
7.3.10	函数 CMP1_POSITIVE_INPUT_SELECT.....	173
7.3.11	函数 CMP1_NEGATIVE_INPUT_SELECT.....	174
7.3.12	函数 CMP2_POSITIVE_INPUT_SELECT.....	174
7.3.13	函数 CMP2_NEGATIVE_INPUT_SELECT.....	175
7.3.14	函数 CMP3_POSITIVE_INPUT_SELECT.....	175
7.3.15	函数 CMP3_NEGATIVE_INPUT_SELECT.....	176
7.3.16	函数 CMP0_OUTPUT_POL_SELECT.....	176
7.3.17	函数 CMP1_OUTPUT_POL_SELECT.....	176
7.3.18	函数 CMP2_OUTPUT_POL_SELECT.....	177
7.3.19	函数 CMP3_OUTPUT_POL_SELECT.....	177
7.3.20	函数 CMP_OUTPUT_SELECT.....	177
7.3.21	函数 CMP0_Read_Output_State.....	177
7.3.22	函数 CMP1_Read_Output_State.....	178
7.3.23	函数 CMP2_Read_Output_State.....	178
7.3.24	函数 CMP3_Read_Output_State.....	178
7.3.25	函数 CMP0_Get_Update_INT_Flag.....	178
7.3.26	函数 CMP1_Get_Update_INT_Flag.....	179
7.3.27	函数 CMP2_Get_Update_INT_Flag.....	179
7.3.28	函数 CMP3_Get_Update_INT_Flag.....	179
7.3.29	函数 CMP_Trigger_Select_Config.....	179
7.3.30	函数 CMP0_Clear_Trigger_INT_Flag.....	180
7.3.31	函数 CMP1_Clear_Trigger_INT_Flag.....	180
7.3.32	函数 CMP2_Clear_Trigger_INT_Flag.....	180
7.3.33	函数 CMP3_Clear_Trigger_INT_Flag.....	180
7.3.34	函数 CMP0_INT_Enable.....	180
7.3.35	函数 CMP1_INT_Enable.....	181
7.3.36	函数 CMP2_INT_Enable.....	181
7.3.37	函数 CMP3_INT_Enable.....	181
7.3.38	函数 CMP_SluggishVoltage_Select.....	182
7.3.39	函数 CMP_HALLMODE_Select.....	182
7.3.40	函数 CMP_BEMF_Enable.....	182
7.3.41	函数 CMP_FLTINSEL_Select.....	182
8	循环冗余校验单元 (CRC)	184
8.1	文件引用关系.....	184
8.2	CRC 寄存器结构.....	184
8.3	CRC 宏定义.....	185
8.4	CRC 库函数.....	186
8.4.1	函数 CRC_Reset.....	186

8.4.2	函数 CRC_Configuration.....	186
8.4.3	函数 CRC_Struct_Init.....	187
8.4.4	函数 CRC_INPUT_DATA.....	187
8.4.5	函数 CRC_GET_RESULT.....	187
8.4.6	函数 CRC_SET_INITVALUE.....	187
8.4.7	函数 CRC_SET_PLN.....	188
8.4.8	函数 CRC_SET_RXOR.....	188
8.4.9	函数 CRC_SET_IDATA.....	188
8.4.10	函数 CRC_GET_TEMP.....	188
8.4.11	函数 CRC_SET_RSET.....	189
9	直接存储器 (DMA)	190
9.1	文件引用关系.....	190
9.2	DMA 寄存器结构.....	190
9.3	DMA 库函数.....	194
9.3.1	函数 DMA_Reset.....	195
9.3.2	函数 DMA_Configuration.....	196
9.3.3	函数 DMA_Struct_Init.....	196
9.3.4	函数 DMA_Transfer_Number_Config.....	196
9.3.5	函数 DMA_Memory_To_Memory_Enable.....	197
9.3.6	函数 DMA_Channel_Priority_Config.....	197
9.3.7	函数 DMA_Peripheral_Data_Width_Config.....	198
9.3.8	函数 DMA_Memory_Data_Width_Config.....	198
9.3.9	函数 DMA_Peripheral_addr_increase_Enable.....	199
9.3.10	函数 DMA_Memory_addr_increase_Enable.....	199
9.3.11	函数 DMA_Loop_Mode_Enable.....	200
9.3.12	函数 DMA_Transfer_Direction_Config.....	200
9.3.13	函数 DMA_Transfer_Mode_Config.....	201
9.3.14	函数 DMA_Oneshot_Enable.....	202
9.3.15	函数 DMA_Channel_Enable.....	202
9.3.16	函数 DMA_Peripheral_Start_Address_Config.....	203
9.3.17	函数 DMA_Memory_Start_Address_Config.....	203
9.3.18	函数 DMA_Get_Peripheral_Current_Address.....	204
9.3.19	函数 DMA_Get_Memory_Current_Address.....	204
9.3.20	函数 DMA_Get_Transfer_Number_Remain.....	205
9.3.21	函数 DMA_Get_INT_Flag.....	205
9.3.22	函数 DMA_Clear_INT_Flag.....	206
9.3.23	函数 DMA_Set_INT_Enable.....	206
9.3.24	函数 DMA_Get_Error_Transfer_INT_Flag.....	207
9.3.25	函数 DMA_Get_Half_Transfer_INT_Flag.....	207
9.3.26	函数 DMA_Get_Finish_Transfer_INT_Flag.....	208
9.3.27	函数 DMA_Error_Transfer_INT_Enable.....	208
9.3.28	函数 DMA_Half_Transfer_INT_Enable.....	209
9.3.29	函数 DMA_Finish_Transfer_INT_Enable.....	209

10 闪存 (FLASH)	211
10.1 文件引用关系.....	211
10.2 FLASH 寄存器结构.....	211
10.3 FLASH 库函数.....	213
10.3.1 函数 CHECK_RESTRICTION_RAM.....	215
10.3.2 函数 SFR_Config_RAM.....	216
10.3.3 函数 FLASH_Get_NonVolatile_Memory_Unlock_Status_RAM.....	216
10.3.4 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM.....	216
10.3.5 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status.....	217
10.3.6 函数 FLASH_Unlock_ISP_RAM.....	217
10.3.7 函数 FLASH_Unlock_ISP.....	217
10.3.8 函数 FLASH_Get_Flash_Unlock_Status_RAM.....	217
10.3.9 函数 FLASH_Clear_Flash_Unlock_Status_RAM.....	218
10.3.10 函数 FLASH_Clear_Flash_Unlock_Status.....	218
10.3.11 函数 FLASH_Unlock_Code_RAM.....	218
10.3.12 函数 FLASH_Unlock_Code.....	219
10.3.13 函数 FLASH_Get_Config_Unlock_Status_RAM.....	219
10.3.14 函数 FLASH_Clear_Config_Unlock_Status_RAM.....	219
10.3.15 函数 FLASH_Clear_Config_Unlock_Status.....	219
10.3.16 函数 FLASH_Unlock_User_Config_RAM.....	220
10.3.17 函数 FLASH_Unlock_User_Config.....	220
10.3.18 函数 FLASH_Data_Write_Enable_RAM.....	220
10.3.19 函数 FLASH_Zone_Config_RAM.....	221
10.3.20 函数 FLASH_Zone_Config.....	221
10.3.21 函数 FLASH_Standby_Mode_Config_RAM.....	221
10.3.22 函数 FLASH_Read_Mode_Config_RAM.....	222
10.3.23 函数 FLASH_Calibration_Update_Enable_RAM.....	222
10.3.24 函数 FLASH_Information_Zone_Wipe_Unlock_Config_RAM.....	222
10.3.25 函数 FLASH_Half_Page_Write_Size_Config_RAM.....	223
10.3.26 函数 FLASH_Program_Cmd_Config_RAM.....	223
10.3.27 函数 FLASH_Executive_Cmd_RAM.....	223
10.3.28 函数 FLASH_Executive_Cmd.....	224
10.3.29 函数 FLASH_NonVolatile_Memory_ECC_Enable_RAM.....	224
10.3.30 函数 FLASH_Linear_Prefetch_Enable_RAM.....	224
10.3.31 函数 FLASH_Period_Number_Config_RAM.....	224
10.3.32 函数 FLASH_Program_Addr_Config_RAM.....	225
10.3.33 函数 FLASH_Get_Program_Status_RAM.....	225
10.3.34 函数 FLASH_Get_Program_Status.....	225
10.3.35 函数 FLASH_Get_Wipe_Status_RAM.....	226
10.3.36 函数 FLASH_Get_Wipe_Status.....	226
10.3.37 函数 FLASH_Get_Compute_Complete_Status_RAM.....	226
10.3.38 函数 FLASH_Clear_Compute_Complete_Status_RAM.....	226
10.3.39 函数 FLASH_Get_CFG_Error_Flag_RAM.....	227

10.3.40	函数 FLASH_Clear_CFG_Error_Flag_RAM.....	227
10.3.41	函数 FLASH_CheckSum_Addr_Config_RAM.....	227
10.3.42	函数 FLASH_Start_SIG_Compute_Enable_RAM.....	228
10.3.43	函数 FLASH_Get_CheckSum_Result_RAM.....	228
10.3.44	函数 FLASH_Wipe_Configuration_RAM.....	228
10.3.45	函数 FLASH_Wipe_Configuration.....	229
10.3.46	函数 FLASH_Program_Configuration_RAM.....	229
10.3.47	函数 FLASH_Program_Configuration.....	230
10.3.48	函数 Read_Flash_or_CFR_RAM.....	230
10.3.49	函数 Read_Flash_or_CFR.....	231
10.3.50	函数 Read_Soft_Device_ID1.....	231
10.3.51	函数 Read_Soft_Device_ID2.....	232
10.3.52	函数 Read_Soft_Device_ID3.....	232
10.3.53	函数 Read_Soft_Device_ID4.....	232
11	通用输入/输出引脚 (GPIO)	错误！未定义书签。
11.1	文件引用关系.....	错误！未定义书签。
11.2	GPIO 寄存器结构.....	错误！未定义书签。
11.3	GPIO 库函数.....	错误！未定义书签。
11.3.1	函数 GPIO_Reset.....	错误！未定义书签。
11.3.2	函数 GPIO_Configuration.....	错误！未定义书签。
11.3.3	函数 GPIO_Struct_Init.....	错误！未定义书签。
11.3.4	函数 GPIO_Pin_Lock_Config.....	错误！未定义书签。
11.3.5	函数 GPIO_Pull_Up_Enable.....	错误！未定义书签。
11.3.6	函数 GPIO_Toggle_Pull_Up_Config.....	错误！未定义书签。
11.3.7	函数 GPIO_Pull_Down_Enable.....	错误！未定义书签。
11.3.8	函数 GPIO_Toggle_Pull_Down_Config.....	错误！未定义书签。
11.3.9	函数 GPIO_Open_Drain_Enable.....	错误！未定义书签。
11.3.10	函数 GPIO_Toggle_Open_Drain_Config.....	错误！未定义书签。
11.3.11	函数 GPIO_Write_Mode_Bits.....	错误！未定义书签。
11.3.12	函数 GPIO_Speed_Config.....	错误！未定义书签。
11.3.13	函数 GPIO_Toggle_Speed_Config.....	错误！未定义书签。
11.3.14	函数 GPIO_Read_Input_Data_Bit.....	错误！未定义书签。
11.3.15	函数 GPIO_Read_Input_Data.....	错误！未定义书签。
11.3.16	函数 GPIO_Read_Output_Data_Bit.....	错误！未定义书签。
11.3.17	函数 GPIO_Read_Output_Data.....	错误！未定义书签。
11.3.18	函数 GPIO_Set_Output_Data_Bits.....	错误！未定义书签。
11.3.19	函数 GPIO_Toggle_Output_Data_Config.....	错误！未定义书签。
11.3.20	函数 GPIO_Pin_RMP_Config.....	错误！未定义书签。
12	通用定时/计数器 (GPTIM)	233
12.1	文件引用关系.....	244
12.2	GPTIM 寄存器结构.....	244
12.3	GPTIM 宏定义.....	246

12.4	GPTIM 库函数.....	246
12.4.1	函数 GPTIM_Configuration.....	247
12.4.2	函数 GPTIM_Struct_Init.....	248
12.4.3	函数 GPTIM_Cmd.....	248
12.4.4	函数 GPTIM_Set_Counter.....	248
12.4.5	函数 GPTIM_Set_Period.....	249
12.4.6	函数 GPTIM_Set_Prescaler.....	249
12.4.7	函数 GPTIM_Counter_Mode_Config.....	249
12.4.8	函数 GPTIM_Clock_Config.....	250
12.4.9	函数 GPTIM_External_Pulse_Sync_Config.....	250
12.4.10	函数 GPTIM_Work_Mode_Config.....	251
12.4.11	函数 GPTIM_Update_Immediately_Config.....	251
12.4.12	函数 GPTIM_Master_Slave_Sync_Config.....	251
12.4.13	函数 GPTIM_Trigger_Select_Config.....	252
12.4.14	函数 GPTIM_Slave_Mode_Config.....	252
12.4.15	函数 GPTIM_Master_Mode_Config.....	253
12.4.16	函数 GPTIM_Update_Rising_Edge_Config.....	253
12.4.17	函数 GPTIM_Update_Enable.....	254
12.4.18	函数 GPTIM_Trigger_DMA_Enable.....	254
12.4.19	函数 GPTIM_Update_DMA_Enable.....	255
12.4.20	函数 GPTIM_Update_INT_Enable.....	255
12.4.21	函数 GPTIM_Trigger_INT_Enable.....	255
12.4.22	函数 GPTIM_Generate_Trigger_Config.....	256
12.4.23	函数 GPTIM_Get_Direction.....	256
12.4.24	函数 GPTIM_Get_Counter.....	257
12.4.25	函数 GPTIM_Get_Period.....	257
12.4.26	函数 GPTIM_Get_Prescaler.....	257
12.4.27	函数 GPTIM_Overflow_INT_Enable.....	258
12.4.28	函数 GPTIM_Clear_Overflow_INT_Flag.....	258
12.4.29	函数 GPTIM_Clear_Update_INT_Flag.....	258
12.4.30	函数 GPTIM_Clear_Trigger_INT_Flag.....	259
12.4.31	函数 GPTIM_Get_Overflow_INT_Flag.....	259
12.4.32	函数 GPTIM_Get_Update_INT_Flag.....	259
12.4.33	函数 GPTIM_Get_Trigger_INT_Flag.....	260
12.4.34	函数 GPTIM_Get_Update_DMA_INT_Flag.....	260
12.4.35	函数 GPTIM_Get_Trigger_DMA_INT_Flag.....	260
13	内部集成电路接口 (I2C)	262
13.1	文件引用关系.....	262
13.2	I2C 寄存器结构.....	262
13.3	I2C 库函数.....	264
13.3.1	函数 I2C_Reset.....	265
13.3.2	函数 I2C_Configuration.....	266
13.3.3	函数 I2C_Struct_Init.....	266

13.3.4	函数 I2C_Cmd.....	266
13.3.5	函数 I2C_Bufr_Address_Config.....	267
13.3.6	函数 I2C_Generate_START.....	267
13.3.7	函数 I2C_Generate_STOP.....	267
13.3.8	函数 I2C_Ack_Config.....	267
13.3.9	函数 I2C_Ack_DATA_Config.....	268
13.3.10	函数 I2C_Call_Cmd.....	268
13.3.11	函数 I2C_Clock_Config.....	268
13.3.12	函数 I2C_MATCH_ADDRESS_Config.....	269
13.3.13	函数 I2C_SCL_Enable.....	269
13.3.14	函数 I2C_NMENA_Enable.....	269
13.3.15	函数 I2C_SMBUS_Enable.....	270
13.3.16	函数 I2C_SMBT_Config.....	270
13.3.17	函数 I2C_SMBus_ALERT_Config.....	270
13.3.18	函数 I2C_SendData.....	271
13.3.19	函数 I2C_SendData8.....	271
13.3.20	函数 I2C_ReceiveData.....	271
13.3.21	函数 I2C_ARP_Enable.....	271
13.3.22	函数 I2C_ADDR_Config.....	272
13.3.23	函数 I2C_MSK_Config.....	272
13.3.24	函数 I2C_BRGH_Config.....	272
13.3.25	函数 I2C_BRGL_Config.....	273
13.3.26	函数 I2C_Start_INT_Enable.....	273
13.3.27	函数 I2C_Stop_INT_Enable.....	273
13.3.28	函数 I2C_Ack_Fail_INT_Enable.....	274
13.3.29	函数 I2C_Arbitration_Lost_INT_Enable.....	274
13.3.30	函数 I2C_SMBus_Alert_INT_Enable.....	274
13.3.31	函数 I2C_SMBus_HostHead_INT_Enable.....	274
13.3.32	函数 I2C_SMBus_Device_Defaultaddress_INT_Enable.....	275
13.3.33	函数 I2C_ISIE_INT_Enable.....	275
13.3.34	函数 I2C_Receive_DMA_INT_Enable.....	275
13.3.35	函数 I2C_Transmit_DMA_INT_Enable.....	276
13.3.36	函数 I2C_Get_Start_Flag.....	276
13.3.37	函数 I2C_Clear_Start_Flag.....	276
13.3.38	函数 I2C_Get_Stop_Flag.....	277
13.3.39	函数 I2C_Clear_Stop_Flag.....	277
13.3.40	函数 I2C_Get_Address_Match_Flag.....	277
13.3.41	函数 I2C_Get_HighAddress_Flag.....	277
13.3.42	函数 I2C_Get_Data_Flag.....	278
13.3.43	函数 I2C_Get_Ack_Fail_Flag.....	278
13.3.44	函数 I2C_Clear_Ack_Fail_Flag.....	278
13.3.45	函数 I2C_Get_Arbitration_Lost_Flag.....	279
13.3.46	函数 I2C_Get_Write_Read_Flag.....	279
13.3.47	函数 I2C_Get_SMBus_Alert_Flag.....	279

13.3.48	函数 I2C_Clear_SMBus_Alert_Flag.....	280
13.3.49	函数 I2C_Get_SMBus_Host_Header_Flag.....	280
13.3.50	函数 I2C_Clear_SMBus_Host_Header_Flag.....	280
13.3.51	函数 I2C_Get_SMBus_Device_Default_Flag.....	280
13.3.52	函数 I2C_Clear_SMBus_Device_Default_Flag.....	281
13.3.53	函数 I2C_Get_INTERRUPT_Flag.....	281
13.3.54	函数 I2C_Clear_INTERRUPT_Flag.....	281
13.3.55	函数 I2C_Get_Receive_Buff_Flag.....	282
13.3.56	函数 I2C_Get_Transmit_Buff_Flag.....	282
13.3.57	函数 I2C_Get_Receive_DMA_Flag.....	282
13.3.58	函数 I2C_Get_Transmit_DMA_Flag.....	282
14	中断 (INT)	284
14.1	文件引用关系.....	284
14.2	INT 寄存器结构.....	284
14.3	INT 宏定义.....	287
14.4	INT 库函数.....	287
14.4.1	函数 INT_Get_Interrupt_Action.....	288
14.4.2	函数 INT_Get_Priority_Pending_Action.....	288
14.4.3	函数 INT_Priority_Base.....	288
14.4.4	函数 INT_Get_Priority_Base.....	289
14.4.5	函数 INT_Stack_Align_Config.....	289
14.4.6	函数 INT_Fault_Masking_Config.....	289
14.4.7	函数 INT_Get_Pre_Empty.....	289
14.4.8	函数 INT_Get_Pending_Flag.....	290
14.4.9	函数 INT_Priority_Group_Config.....	290
14.4.10	函数 INT_Get_Priority_Group.....	290
14.4.11	函数 INT_All_Enable.....	291
14.4.12	函数 INT_Interrupt_Enable.....	291
14.4.13	函数 INT_Set_Systick_Flag.....	291
14.4.14	函数 INT_Set_PendSV_Flag.....	291
14.4.15	函数 INT_Get_Interrupt_Flag.....	292
14.4.16	函数 INT_Clear_Interrupt_Flag.....	292
14.4.17	函数 INT_Interrupt_Priority_Config.....	292
14.4.18	函数 INT_Set_Interrupt_Priority.....	293
14.4.19	函数 INT_Stack_Delay_Enable.....	293
14.4.20	函数 INT_External_Configuration.....	293
14.4.21	函数 INT_External_Struct_Init.....	294
14.4.22	函数 INT_External_Mask_Enable.....	294
14.4.23	函数 INT_External_Rise_Enable.....	294
14.4.24	函数 INT_External_Fall_Enable.....	295
14.4.25	函数 INT_Get_External_Flag.....	295
14.4.26	函数 INT_External_Clear_Flag.....	295
14.4.27	函数 INT_External_Source_Enable.....	296

15	独立看门狗 (IWDG)	297
15.1	文件引用关系	297
15.2	IWDG 寄存器结构	297
15.3	IWDG 宏定义	298
15.4	IWDG 库函数	298
15.4.1	函数 IWDG_Prescaler_Config	298
15.4.2	函数 IWDG_Overflow_Config	298
15.4.3	函数 IWDG_Enable	299
15.4.4	函数 IWDG_Feed_The_Dog	299
16	可编程增益放大器模块 (OP)	300
16.1	文件引用关系	300
16.2	OP 寄存器结构	300
16.3	OP 库函数	301
16.3.1	函数 OP_Reset	301
16.3.2	函数 OP_CAL_Configure	301
16.3.3	函数 OP_GAIN_SELECT	302
16.3.4	函数 OP3_POSITIVE_INPUT_SELECT	302
16.3.5	函数 OP_OUTPUT_EN	302
16.3.6	函数 OP_MODULE_EN	302
17	振荡器 (OSC)	304
17.1	文件引用关系	304
17.2	OSC 寄存器结构	304
17.3	OSC 宏定义	305
17.4	OSC 库函数	305
17.4.1	函数 OSC_SCLK_Configuration	307
17.4.2	函数 OSC_HFCK_Configuration	307
17.4.3	函数 OSC_LFCK_Configuration	307
17.4.4	函数 OSC_CK48M_Configuration	308
17.4.5	函数 OSC_Struct_Init	308
17.4.6	函数 OSC_LFCK_Division_Config	308
17.4.7	函数 OSC_HFCK_Division_Config	309
17.4.8	函数 OSC_SCK_Division_Config	309
17.4.9	函数 OSC_PLL_Input_Source_Config	310
17.4.10	函数 OSC_HFCK_Source_Config	310
17.4.11	函数 OSC_HFCK_Enable	310
17.4.12	函数 OSC_LFCK_Source_Config	311
17.4.13	函数 OSC_LFCK_Enable	311
17.4.14	函数 OSC_SCK_Source_Config	311
17.4.15	函数 OSC_Backup_Write_Read_Enable	312
17.4.16	函数 OSC_SCLK_Output_Enable	312
17.4.17	函数 OSC_SCLK_Output_Select	312

17. 4. 18	函数 OSC_SCLK_Output_Division_Config.....	313
17. 4. 19	函数 OSC_Clock_Failure_Check_Enable.....	313
17. 4. 20	函数 OSC_CK48M_Division_Config.....	313
17. 4. 21	函数 OSC_CK48M_Source_Config.....	314
17. 4. 22	函数 OSC_CK48M_Enable.....	314
17. 4. 23	函数 OSC_PLL_Multiple_Value_Select.....	314
17. 4. 24	函数 OSC_PLL_RST.....	315
17. 4. 25	函数 OSC_PLL_Start_Delay_Config.....	315
17. 4. 26	函数 OSC_EXTHF_Start_Delay_Config.....	315
17. 4. 27	函数 OSC_EXTLF_Start_Delay_Config.....	316
17. 4. 28	函数 OSC_PLL_Software_Enable.....	316
17. 4. 29	函数 OSC_EXTHF_Software_Enable.....	317
17. 4. 30	函数 OSC_EXTLF_Software_Enable.....	317
17. 4. 31	函数 OSC_INTHF_Software_Enable.....	317
17. 4. 32	函数 OSC_INTLF_Software_Enable.....	317
17. 4. 33	函数 OSC_Zero_Drift_Config.....	318
17. 4. 34	函数 OSC_Positive_Drift_Config.....	318
17. 4. 35	函数 OSC_Negative_Drift_Config.....	319
17. 4. 36	函数 OSC_Current_Gain_Config.....	319
17. 4. 37	函数 OSC_High_Speed_Enable.....	319
17. 4. 38	函数 OSC_External_Input_Enable.....	319
17. 4. 39	函数 OSC_Feedback_Resistance_Config.....	320
17. 4. 40	函数 OSC_PLL_Zero_Source_Enable.....	320
17. 4. 41	函数 OSC_PLL_Vref2_Enable.....	320
17. 4. 42	函数 OSC_PLL_Vref0p5_Enable.....	321
17. 4. 43	函数 OSC_PLL_Vref1p2_Enable.....	321
17. 4. 44	函数 OSC_PLL_Low_Power_20nA_Enable.....	321
17. 4. 45	函数 OSC_PLL_Vref1p14_Enable.....	321
17. 4. 46	函数 OSC_PLL_Low_Power_100nA_Enable.....	322
17. 4. 47	函数 OSC_PLL_LDO_Enable.....	322
17. 4. 48	函数 OSC_PLL_INT_Enable.....	322
17. 4. 49	函数 OSC_EXTHF_INT_Enable.....	323
17. 4. 50	函数 OSC_EXTLF_INT_Enable.....	323
17. 4. 51	函数 OSC_INTHF_INT_Enable.....	323
17. 4. 52	函数 OSC_INTLF_INT_Enable.....	323
17. 4. 53	函数 OSC_Get_Clock_Failure_INT_Flag.....	324
17. 4. 54	函数 OSC_Get_PLL_INT_Flag.....	324
17. 4. 55	函数 OSC_Get_EXTHF_INT_Flag.....	324
17. 4. 56	函数 OSC_Get_EXTLF_INT_Flag.....	324
17. 4. 57	函数 OSC_Get_INTHF_INT_Flag.....	325
17. 4. 58	函数 OSC_Get_INTLF_INT_Flag.....	325
17. 4. 59	函数 OSC_Get_LP4MIF_INT_Flag.....	325
18	外设模块时钟使能模块 (PCLK)	326

18.1	文件引用关系.....	326
18.2	PCLK 寄存器结构.....	326
18.3	PCLK 宏定义.....	327
18.4	PCLK 库函数.....	327
18.4.1	函数 PCLK_CTL0_Peripheral_Clock_Enable.....	327
18.4.2	函数 PCLK_CTL1_Peripheral_Clock_Enable.....	327
18.4.3	函数 PCLK_CTL2_Peripheral_Clock_Enable.....	328
18.4.4	函数 PCLK_CTL3_Peripheral_Clock_Enable.....	329
19	电源管理 (PM)	331
19.1	文件引用关系.....	331
19.2	PM 寄存器结构.....	331
19.3	PM 宏定义.....	332
19.4	PM 库函数.....	332
19.4.1	函数 PM_IO_Latch_Enable.....	334
19.4.2	函数 PM_Get_IO_Latch_Status.....	334
19.4.3	函数 PM_Internal_Low_Frequency_Enable.....	335
19.4.4	函数 PM_External_Low_Frequency_Enable.....	335
19.4.5	函数 PM_External_Low_Frequency_Clock_Enable.....	335
19.4.6	函数 PM_Main_Bandgap_Enable.....	336
19.4.7	函数 PM_LDO18_Enable.....	336
19.4.8	函数 PM_Backup_Registers_Reset_Config.....	336
19.4.9	函数 PM_Independent_Watchdog_Reset_Config.....	336
19.4.10	函数 PM_SRAMA_In_Standby_Work_Mode_Config.....	337
19.4.11	函数 PM_LPRAM_In_Standby_Work_Mode_Config.....	337
19.4.12	函数 PM_Backup_POR_Delay_Time_Config.....	337
19.4.13	函数 PM_Main_POR_Delay_Time_Config.....	338
19.4.14	函数 PM_Peripheral_IO_Port_Config.....	338
19.4.15	函数 PM_OCAL0LOCK_Enable.....	338
19.4.16	函数 PM_MEMSEL_Enable.....	339
19.4.17	函数 PM_Flash_Power_Off_Enable.....	339
19.4.18	函数 PM_CCP0CLKLPEN_Enable.....	339
19.4.19	函数 PM_Backup_Write_And_Read_Enable.....	339
19.4.20	函数 PM_VREF_Software_Enable.....	340
19.4.21	函数 PM_VREF_SELECT.....	340
19.4.22	函数 PM_LPR_Software_Enable.....	340
19.4.23	函数 PM_Low_Power_Mode_Config.....	340
19.4.24	函数 PM_BOR_Enable.....	341
19.4.25	函数 PM_Low_Power_BOR_Enable.....	341
19.4.26	函数 PM_Temperature_Sensor_Enable.....	341
19.4.27	函数 PM_Temperature_Sensor_Buffer_Enable.....	342
19.4.28	函数 PM_Reference_Voltage_Enable.....	342
19.4.29	函数 PM_Internal_Test_Buffer_Clock_Enable.....	342
19.4.30	函数 PM_Internal_Test_Buffer_Clock_Scaler_Config.....	342

19. 4. 31	函数 PM_PLL0_Output_Buffer_Enable.....	343
19. 4. 32	函数 PM_PLL1_Output_Buffer_Enable.....	343
19. 4. 33	函数 PM_PLL2_Output_Buffer_Enable.....	343
19. 4. 34	函数 PM_PLL0LDO_Output_Buffer_Enable.....	344
19. 4. 35	函数 PM_PLL1LDO_Output_Buffer_Enable.....	344
19. 4. 36	函数 PM_PLL2LDO_Output_Buffer_Enable.....	344
19. 4. 37	函数 PM_Battery_BOR_Config.....	344
19. 4. 38	函数 PM_Battery_BOR_Enable.....	345
19. 4. 39	函数 PM_Peripheral_Voltage_Monitoring_Enable.....	345
19. 4. 40	函数 PM_Voltage_Detection_Config.....	345
19. 4. 41	函数 PM_Voltage_Detection_Enable.....	346
19. 4. 42	函数 PM_External_Wakeup_Pin_Enable.....	346
19. 4. 43	函数 PM_External_Wakeup_Edge_Config.....	346
19. 4. 44	函数 PM_Stop_Mode_Peripheral_INLF_Enable.....	347
19. 4. 45	函数 PM_Peripheral_Reset_Config.....	347
19. 4. 46	函数 PM_Vdd_Por_Enable.....	348
19. 4. 47	函数 PM_Low_Power_Bandgap_Enable.....	348
19. 4. 48	函数 PM_Power_Dissipation_Mode_Config.....	348
19. 4. 49	函数 PM_Power_Dissipation_Mode_Delay_Config.....	348
19. 4. 50	函数 PM_Internal_Test_Buffer_Enable.....	349
19. 4. 51	函数 PM_Clear_Reset_And_Wakeup_Flag.....	349
19. 4. 52	函数 PM_Get_IWDT_Reset_Flag.....	349
19. 4. 53	函数 PM_Clear_External_Wakeup_Pin_Flag.....	350
19. 4. 54	函数 PM_Get_Low_Power_Running_State.....	350
19. 4. 55	函数 PM_Get_LPR_Status.....	351
19. 4. 56	函数 PM_Get_Peripheral_Voltage_Detection_Status.....	351
19. 4. 57	函数 PM_Zero_Drift_Current_Config.....	351
19. 4. 58	函数 PM_BOR_Voltage_Config.....	352
19. 4. 59	函数 PM_Main_Regulator_Voltage_Config.....	352
19. 4. 60	函数 PM_Main_Regulator_HV_Enable.....	352
19. 4. 61	函数 PM_Reference_Calibration_Config.....	353
19. 4. 62	函数 PM_INTLF_Bias_Current_Config.....	353
19. 4. 63	函数 PM_EXTLF_Bias_Current_Config.....	354
19. 4. 64	函数 PM_INTLF_Capacitance_Calibration_Config.....	354
19. 4. 65	函数 PM_LP_Bias_Calibration_Config.....	354
19. 4. 66	函数 PM_LPBGPump_Calibration_Config.....	355
19. 4. 67	函数 PM_EXTLF_N_Bias_Current_Config.....	355
19. 4. 68	函数 PM_EXTLF_PIN_Selection_Config.....	355
19. 4. 69	函数 PM_EXTHF_PIN_Selection_Config.....	356
19. 4. 70	函数 PM_LDO18_Module_Config.....	356
19. 4. 71	函数 PM_Main_Regulator_Bandgap_Config.....	356
19. 4. 72	函数 PM_LPR_Module_Config.....	356
20	实时时钟 (RTC)	358

20.1	文件引用关系.....	358
20.2	RTC 寄存器结构.....	358
20.3	RTC 库函数.....	360
20.3.1	函数 RTC_Reset.....	363
20.3.2	函数 RTC_Configuration.....	363
20.3.3	函数 RTC_Time_Struct_Init.....	364
20.3.4	函数 RTC_Date_Struct_Init.....	364
20.3.5	函数 RTC_Struct_Init.....	364
20.3.6	函数 RTC_Get_Time_Configuration.....	364
20.3.7	函数 RTC_Get_Date_Configuration.....	365
20.3.8	函数 RTC_Alarm_Configuration.....	365
20.3.9	函数 RTC_Alarm_Struct_Init.....	365
20.3.10	函数 RTC_Clock_Calibration_Config.....	366
20.3.11	函数 RTC_Time_Tick_Output_Enable.....	366
20.3.12	函数 RTC_Time_Stamp_Edge_Config.....	366
20.3.13	函数 RTC_Time_Stamp_Edge_Enable.....	367
20.3.14	函数 RTC_Add_One_Hour_Enable.....	367
20.3.15	函数 RTC_Sub_One_Hour_Enable.....	367
20.3.16	函数 RTC_Time_Tick_Config.....	367
20.3.17	函数 RTC_Start_Config.....	368
20.3.18	函数 RTC_Reset_Config.....	368
20.3.19	函数 RTC_Get_Leap_Year_Flag.....	368
20.3.20	函数 RTC_Hour_Format_Config.....	369
20.3.21	函数 RTC_Config_Mode_Enable.....	369
20.3.22	函数 RTC_Get_Operation_Off_Flag.....	369
20.3.23	函数 RTC_Get_Action_State_Flag.....	369
20.3.24	函数 RTC_Enable.....	370
20.3.25	函数 RTC_Alarm_A_Enable.....	370
20.3.26	函数 RTC_Alarm_A_Weekday_Enable.....	370
20.3.27	函数 RTC_Alarm_A_Weekday_Config.....	370
20.3.28	函数 RTC_Alarm_A_Hours_Enable.....	371
20.3.29	函数 RTC_Alarm_A_AMPM_Config.....	371
20.3.30	函数 RTC_Alarm_A_Hours_Config.....	371
20.3.31	函数 RTC_Alarm_A_Minutes_Enable.....	372
20.3.32	函数 RTC_Alarm_A_Minutes_Config.....	372
20.3.33	函数 RTC_Alarm_A_Seconds_Enable.....	372
20.3.34	函数 RTC_Alarm_A_Seconds_Config.....	372
20.3.35	函数 RTC_Alarm_B_Enable.....	373
20.3.36	函数 RTC_Alarm_B_Weekday_Enable.....	373
20.3.37	函数 RTC_Alarm_B_Weekday_Config.....	373
20.3.38	函数 RTC_Alarm_B_Hours_Enable.....	374
20.3.39	函数 RTC_Alarm_B_AMPM_Config.....	374
20.3.40	函数 RTC_Alarm_B_Hours_Config.....	374
20.3.41	函数 RTC_Alarm_B_Minutes_Enable.....	374

20.3.42	函数 RTC_Alarm_B_Minutes_Config.....	375
20.3.43	函数 RTC_Alarm_B_Seconds_Enable.....	375
20.3.44	函数 RTC_Alarm_B_Seconds_Config.....	375
20.3.45	函数 RTC_Weekday_Config.....	375
20.3.46	函数 RTC_AMPM_Config.....	376
20.3.47	函数 RTC_Hours_Config.....	376
20.3.48	函数 RTC_Minutes_Config.....	376
20.3.49	函数 RTC_Seconds_Config.....	377
20.3.50	函数 RTC_Year_Config.....	377
20.3.51	函数 RTC_Month_Config.....	377
20.3.52	函数 RTC_Day_Config.....	378
20.3.53	函数 RTC_Weekday_Backup_Config.....	378
20.3.54	函数 RTC_AMPM_Backup_Config.....	378
20.3.55	函数 RTC_Hours_Backup_Config.....	379
20.3.56	函数 RTC_Minutes_Backup_Config.....	379
20.3.57	函数 RTC_Seconds_Backup_Config.....	379
20.3.58	函数 RTC_Year_Backup_Config.....	380
20.3.59	函数 RTC_Month_Backup_Config.....	380
20.3.60	函数 RTC_Day_Backup_Config.....	380
20.3.61	函数 RTC_Timer1_Config.....	381
20.3.62	函数 RTC_Timer0_Config.....	381
20.3.63	函数 RTC_Timer1_Enable.....	381
20.3.64	函数 RTC_Timer0_Enable.....	381
20.3.65	函数 RTC_Timer1_Source_Config.....	382
20.3.66	函数 RTC_Timer0_Source_Config.....	382
20.3.67	函数 RTC_Time_Stamp_INT_Enable.....	383
20.3.68	函数 RTC_Time_Stamp_INT_Enable.....	383
20.3.69	函数 RTC_Timer1_INT_Enable.....	383
20.3.70	函数 RTC_Timer0_INT_Enable.....	383
20.3.71	函数 RTC_Time_Tick_INT_Enable.....	384
20.3.72	函数 RTC_Alarm_B_INT_Enable.....	384
20.3.73	函数 RTC_Alarm_A_INT_Enable.....	384
20.3.74	函数 RTC_Days_INT_Enable.....	384
20.3.75	函数 RTC_Hours_INT_Enable.....	385
20.3.76	函数 RTC_Minutes_INT_Enable.....	385
20.3.77	函数 RTC_Seconds_INT_Enable.....	385
20.3.78	函数 RTC_Get_Time_Stamp_INT_Flag.....	385
20.3.79	函数 RTC_Get_Time_Stamp_Overflow_INT_Flag.....	386
20.3.80	函数 RTC_Get_Timer1_INT_Flag.....	386
20.3.81	函数 RTC_Get_Timer0_INT_Flag.....	386
20.3.82	函数 RTC_Get_Timer0_INT_Flag.....	386
20.3.83	函数 RTC_Get_Alarm_B_INT_Flag.....	387
20.3.84	函数 RTC_Get_Alarm_A_INT_Flag.....	387
20.3.85	函数 RTC_Get_Days_INT_Flag.....	387

20.3.86	函数 RTC_Get_Hours_INT_Flag.....	388
20.3.87	函数 RTC_Get_Hours_INT_Flag.....	388
20.3.88	函数 RTC_Get_Seconds_INT_Flag.....	388
20.3.89	函数 RTC_Clear_Time_Stamp_INT_Flag.....	388
20.3.90	函数 RTC_Clear_Time_Stamp_Overflow_INT_Flag.....	389
20.3.91	函数 RTC_Clear_Timer1_INT_Flag.....	389
20.3.92	函数 RTC_Clear_Timer0_INT_Flag.....	389
20.3.93	函数 RTC_Clear_Time_Tick_INT_Flag.....	389
20.3.94	函数 RTC_Clear_Alarm_B_INT_Flag.....	390
20.3.95	函数 RTC_Clear_Alarm_A_INT_Flag.....	390
20.3.96	函数 RTC_Clear_Days_INT_Flag.....	390
20.3.97	函数 RTC_Clear_Hours_INT_Flag.....	390
20.3.98	函数 RTC_Clear_Minutes_INT_Flag.....	391
20.3.99	函数 RTC_Clear_Seconds_INT_Flag.....	391
21	串行外设接口 (SPI)	392
21.1	文件引用关系.....	392
21.2	SPI 寄存器结构.....	392
21.3	SPI 宏定义.....	394
21.4	SPI 库函数.....	394
21.4.1	函数 SPI_Reset.....	395
21.4.2	函数 SPI_Configuration.....	395
21.4.3	函数 I2S_Configuration.....	396
21.4.4	函数 SPI_Struct_Init.....	396
21.4.5	函数 I2S_Struct_Init.....	396
21.4.6	函数 SPI_Cmd.....	396
21.4.7	函数 I2S_Mode_Select.....	397
21.4.8	函数 SPI_I2S_ReceiveData.....	397
21.4.9	函数 SPI_I2S_SendData32.....	397
21.4.10	函数 SPI_I2S_SendData8.....	397
21.4.11	函数 SPI_BaudRate_Config.....	398
21.4.12	函数 I2S_DIV_Config.....	398
21.4.13	函数 SPI_MODE_Config.....	398
21.4.14	函数 SPI_CLK_Config.....	399
21.4.15	函数 SPI_Data_Direction_Config.....	399
21.4.16	函数 SPI_Clock_Polarity_Config.....	399
21.4.17	函数 SPI_Clock_Edge_Config.....	400
21.4.18	函数 SPI_BIT_SELECT_Config.....	400
21.4.19	函数 SPI_I2S_MODE_Config.....	400
21.4.20	SPI_I2S_STANDARD_Config.....	401
21.4.21	函数 SPI_PCM_Config.....	401
21.4.22	函数 SPI_CHLEN_Config.....	402
21.4.23	函数 SPI_PCM_CLOCK_Polarity_Config.....	402
21.4.24	函数 SPI_MAIN_CLOCK_OUT_Enable.....	402

21.4.25	函数 SPI_Receive_Overflow_INT_Enable.....	403
21.4.26	函数 SPI_Transmit_Overflow_INT_Enable.....	403
21.4.27	函数 SPI_RNEIE_INT_Enable.....	403
21.4.28	函数 SPI_TNEIE_INT_Enable.....	403
21.4.29	函数 SPI_Receive_DMA_INT_Enable.....	404
21.4.30	函数 SPI_Transmit_DMA_INT_Enable.....	404
21.4.31	函数 SPI_Transmit_CHSIDE_INT_Enable.....	404
21.4.32	函数 SPI_Get_BUSY_Flag.....	405
21.4.33	函数 SPI_Get_Receive_Overflow_Flag.....	405
21.4.34	函数 SPI_Get_Transmit_Overflow_Flag.....	405
21.4.35	函数 SPI_Get_Receive_Buf_Flag.....	405
21.4.36	函数 SPI_Get_Transmit_Buf_Flag.....	406
21.4.37	函数 SPI_Clear_Receive_Overflow_INT_Flag.....	406
21.4.38	函数 SPI_Clear_Transmit_Overflow_INT_Flag.....	406
22	系统控制 (SYSCTL)	407
22.1	文件引用关系.....	407
22.2	SYSCTL 寄存器结构.....	407
22.3	SYSCTL 宏定义.....	408
22.4	SYSCTL 库函数.....	408
22.4.1	函数 SYSCTL_Get_V_Flag.....	409
22.4.2	函数 SYSCTL_Get_C_Flag.....	409
22.4.3	函数 SYSCTL_Get_Z_Flag.....	409
22.4.4	函数 SYSCTL_Get_N_Flag.....	409
22.4.5	函数 SYSCTL_Set_V_Flag.....	410
22.4.6	函数 SYSCTL_Set_C_Flag.....	410
22.4.7	函数 SYSCTL_Set_Z_Flag.....	410
22.4.8	函数 SYSCTL_Set_N_Flag.....	410
22.4.9	函数 SYSCTL_Sleep_On_Exit_Enable.....	411
22.4.10	函数 SYSCTL_Deep_Sleep_Enable.....	411
22.4.11	函数 SYSCTL_Interrupt_Awake_Enable.....	411
22.4.12	函数 SYSCTL_Stack_Align_State.....	411
22.4.13	函数 SYSCTL_Super_User_Config.....	412
22.4.14	函数 SYSCTL_Stack_Pointer_State.....	412
22.4.15	函数 SYSCTL_Stack_Pointer_Config.....	412
22.4.16	函数 SYSCTL_Exception_Reset_Enable.....	412
22.4.17	函数 SYSCTL_System_Reset_Enable.....	413
22.4.18	函数 SYSCTL_Vector_Offset_Config.....	413
22.4.19	函数 SYSCTL_Ram_Space_Config.....	413
22.4.20	函数 SYSCTL_Flash_Start_Remap_Config.....	414
23	节拍定时器 (SYSTICK)	415
23.1	文件引用关系.....	415
23.2	SYSTICK 寄存器结构.....	415

23.3	SYSTICK 宏定义.....	416
23.4	SYSTICK 库函数.....	416
23.4.1	函数 SYSTICK_Configuration.....	416
23.4.2	函数 SYSTICK_Cmd.....	417
23.4.3	函数 SYSTICK_Clock_Config.....	417
23.4.4	函数 SYSTICK_Systick_INT_Enable.....	417
23.4.5	函数 SYSTICK_Get_Count_Zero_Flag.....	418
23.4.6	函数 SYSTICK_Reload_Config.....	418
23.4.7	函数 SYSTICK_Counter_Update.....	418
23.4.8	函数 SYSTICK_Get_Reload.....	419
23.4.9	函数 SYSTICK_Get_Counter.....	419
24	通用同步/异步收发器 (USART)	420
24.1	文件引用关系.....	420
24.2	USART 寄存器结构.....	420
24.3	UASRT 库函数.....	423
24.3.1	函数 USART_Reset.....	426
24.3.2	函数 USART_Configuration.....	426
24.3.3	函数 USART_U7816R_Configuration.....	427
24.3.4	函数 USART_Struct_Init.....	427
24.3.5	函数 USART_U7816R_Struct_Init.....	427
24.3.6	函数 USART_Cmd.....	427
24.3.7	函数 USART_BaudRate_Clock_Config.....	428
24.3.8	函数 USART_HalfDuplex_ClockPolarity_Config.....	428
24.3.9	函数 USART_Transmit_Order_Config.....	429
24.3.10	函数 USART_Receive_Order_Config.....	429
24.3.11	函数 USART_Infrare_Detector_Voltage_Config.....	429
24.3.12	函数 USART_WeakUP_Enable.....	430
24.3.13	函数 USART_Clock_Source_Config.....	430
24.3.14	函数 USART_Address_Detection_Enable.....	430
24.3.15	函数 USART_Auto_BaudRate_Detection_Enable.....	431
24.3.16	函数 USART_Get_Auto_BaudRate_Detection_Flag.....	431
24.3.17	函数 USART_Send_Blank_Enable.....	431
24.3.18	函数 USART_SYN_Choice_Config.....	432
24.3.19	函数 USART_Transmit_Data_Enable.....	432
24.3.20	函数 USART_Receive_Data_Enable.....	432
24.3.21	函数 USART_STOP_Word_Config.....	433
24.3.22	函数 USART_Transmit_9Word_Select_Config.....	433
24.3.23	函数 USART_Parity_Select_Config.....	434
24.3.24	函数 USART_9Data_Enable.....	434
24.3.25	函数 USART_CTS_Enable.....	434
24.3.26	函数 USART_RTS_Enable.....	435
24.3.27	函数 USART_Infrare_Detector_Enable.....	435
24.3.28	函数 USART_RESHD_Enable.....	435

24.3.29	函数 USART_Singlet_Line_Mode_Enable.....	436
24.3.30	函数 USART_BaudRate_Integer_Config.....	436
24.3.31	函数 USART_BaudRate_Decimal1_Config.....	436
24.3.32	函数 USART_BaudRate_Decimal2_Config.....	437
24.3.33	函数 USART_SendData.....	437
24.3.34	函数 USART_TransmitData.....	437
24.3.35	函数 USART_ReceiveData.....	438
24.3.36	函数 USART_Address_Match_Config.....	438
24.3.37	函数 USART_Send_Idle_Frame_Enable.....	438
24.3.38	函数 USART_Receive_Idle_Frame_Config.....	439
24.3.39	函数 USART_7816_Cmd.....	439
24.3.40	函数 USART_7816_CLKOUT_Enable.....	439
24.3.41	函数 USART_7816_Error_Signal_Config.....	440
24.3.42	函数 USART_Passageway_Select_Config.....	440
24.3.43	函数 USART_BGT_Config.....	440
24.3.44	函数 USART_Transmit_Repeat_Enable.....	441
24.3.45	函数 USART_Receive_Repeat_Enable.....	441
24.3.46	函数 USART_Transmit_Repeat_Times_Config.....	442
24.3.47	函数 USART_Receive_Repeat_Times_Config.....	442
24.3.48	函数 USART_7816_CLKDIV_Config.....	442
24.3.49	函数 USART_7816_EGT_Config.....	443
24.3.50	函数 USART_7816_Resend_Mode_Select.....	443
24.3.51	函数 USART_Receive_Overflow_INT_Enable.....	444
24.3.52	函数 USART_Parity_ERROR_INT_Enable.....	444
24.3.53	函数 USART_Frame_ERROE_INT_Enable.....	444
24.3.54	函数 USART_Blank_INT_Enable.....	445
24.3.55	函数 USART_Auto_BaudRate_TimeOver_INT_Enable.....	445
24.3.56	函数 USART_WeakUP_INT_Enable.....	445
24.3.57	函数 USART_Transmit_ERROR_INT_Enable.....	446
24.3.58	函数 USART_Receive_ERROR_INT_Enable.....	446
24.3.59	函数 USART_CTS_INT_Enable.....	446
24.3.60	函数 USART_RDR_INT_Enable.....	447
24.3.61	函数 USART_TFE_INT_Enable.....	447
24.3.62	函数 USART_TXE_INT_Enable.....	447
24.3.63	函数 USART_Receive_DMA_INT_Enable.....	448
24.3.64	函数 USART_Transmit_DMA_INT_Enable.....	448
24.3.65	函数 USART_IDLE_INT_Enable.....	448
24.3.66	函数 USART_UADM_INT_Enable.....	449
24.3.67	函数 USART_Get_Receive_Overflow_Flag.....	449
24.3.68	函数 USART_Get_Parity_ERROR_Flag.....	449
24.3.69	函数 USART_Get_Frame_ERROR_Flag.....	450
24.3.70	函数 USART_Get_Blank_Flag.....	450
24.3.71	函数 USART_Get_Auto_Baudrate_TimeOver_Flag.....	450
24.3.72	函数 USART_Get_WeakUP_Flag.....	451

24.3.73	函数 USART_Get_7816Transmit_ERROR_Flag.....	451
24.3.74	函数 USART_Get_7816Receive_ERROR_Flag.....	451
24.3.75	函数 USART_Get_CTS_Flag.....	452
24.3.76	函数 USART_Get_Receive_BUFR_Ready_Flag.....	452
24.3.77	函数 USART_Get_WUEN_Flag.....	452
24.3.78	函数 USART_Get_Transmit_BUFR_Empty_Flag.....	453
24.3.79	函数 USART_Get_Transmitter_Empty_Flag.....	453
24.3.80	函数 USART_Get_Receive_Frame_Idel_Flag.....	453
24.3.81	函数 USART_Clear_Receive_Overflow_INT_Flag.....	454
24.3.82	函数 USART_Clear_Parity_ERROR_INT_Flag.....	454
24.3.83	函数 USART_Clear_Frame_ERROR_INT_Flag.....	454
24.3.84	函数 USART_Clear_Blank_INT_Flag.....	454
24.3.85	函数 USART_Clear_Auto_BaudRate_TimeOver_INT_Flag.....	455
24.3.86	函数 USART_Clear_WeakUP_INT_Flag.....	455
24.3.87	函数 USART_Clear_Transmit_ERROR_INT_Flag.....	455
24.3.88	函数 USART_Clear_Receive_ERROR_INT_Flag.....	456
24.3.89	函数 USART_Clear_CTS_INT_Flag.....	456
24.3.90	函数 USART_Clear_UADM_INT_Flag.....	456
24.3.91	函数 USART_Clear_IDLE_INT_Flag.....	456
24.3.92	函数 USART_Clear_Receive_BUFR_INT_Flag.....	457
24.3.93	函数 USART_Clear_Transmit_BUFR_INT_Flag.....	457
25	窗口看门狗 (WWDT)	458
25.1	文件引用关系.....	458
25.2	WWDT 寄存器结构.....	458
25.3	WWDT 宏定义.....	458
25.4	WWDT 库函数.....	459
25.4.1	函数 WWDT_Reset.....	459
25.4.2	函数 WWDT_Threshold_Config.....	459
25.4.3	函数 WWDT_Prescaler_Config.....	460
25.4.4	函数 WWDT_Enable.....	460
25.4.5	函数 WWDT_Counter_Config.....	461
25.4.6	函数 WWDT_Get_Counter.....	461
25.4.7	函数 WWDT_INT_Enable.....	461
25.4.8	函数 WWDT_Get_INT_Flag.....	461
25.4.9	函数 WWDT_Clear_INT_Flag.....	462
附录 2	历史使用手册版本信息.....	463

图示列表

图 2-1	ADC 模块文件包含关系.....	54
图 3-1	ATIM/ECCP 模块文件包含关系.....	79
图 4-1	BKP 模块文件包含关系.....	130
图 5-1	BTIM 模块文件包含关系.....	136
图 6-1	IWDT 模块文件包含关系.....	152
图 7-1	CMP 模块文件包含关系.....	168
图 8-1	CRC 模块文件包含关系.....	184
图 9-1	DMA 寄存器结构说明.....	190
图 10-1	FLASH 寄存器结构说明.....	211
图 11-1	GPIO 模块文件包含关系.....	错误！未定义书签。
图 12-1	GPTIM 模块文件包含关系.....	244
图 13-1	I2C 寄存器结构说明.....	262
图 14-1	INT 模块文件包含关系.....	284
图 15-1	IWDT 模块文件包含关系.....	297
图 16-1	OP 模块文件包含关系.....	300
图 17-1	OSC 模块文件包含关系.....	304
图 18-1	PCLK 模块文件包含关系.....	326
图 19-1	PM 模块文件包含关系.....	331
图 20-1	RTC 模块文件包含关系.....	358
图 21-1	SPI 模块文件包含关系.....	392
图 22-1	SYSCTL 模块文件包含关系.....	407
图 23-1	SYSTICK 模块文件包含关系.....	415
图 24-1	USART 寄存器结构说明.....	420
图 25-1	WWDT 模块文件包含关系.....	458

表格列表

表 1-1 外设缩写说明.....	50
表 1-2 外设缩写说明.....	51
表 1-3 固件库目录说明.....	53
表 2-1 ADC 寄存器结构说明.....	55
表 2-2 ADC0 快慢交叉延时寄存器结构说明.....	56
表 2-3 ADC 配置信息结构体说明.....	56
表 2-4 ADC 快慢交叉模式信息结构体说明.....	57
表 2-5 ADC 模拟看门狗信息结构体说明.....	57
表 2-6 ADC 固件库函数列表.....	57
表 2-7 函数 ADC_Reset.....	59
表 2-8 函数 ADC_Configuration.....	59
表 2-9 函数 ADC_Struct_Init.....	60
表 2-10 函数 ADC_Delay_Configuration.....	60
表 2-11 函数 ADC_Delay_Struct_Init.....	60
表 2-12 函数 ADC_Cmd.....	61
表 2-13 函数 ADC_Analog_Watchdog_Configuration.....	61
表 2-14 函数 ADC_Analog_Watchdog_Struct_Init.....	61
表 2-15 函数 ADC_Watchdog_Single_Channel_Enable.....	61
表 2-16 函数 ADC_Scan_Mode_Enable.....	62
表 2-17 函数 ADC_Bosssel_Comparator_Calibration.....	62
表 2-18 函数 ADC_Bosssel_Calibration.....	62
表 2-19 函数 ADC_Trim_Current_Intensity_Bias.....	63
表 2-20 函数 ADC_Analog_Clock_Config.....	63
表 2-21 函数 ADC_Data_Align_Config.....	63
表 2-22 函数 ADC_Clock_Source_Config.....	64
表 2-23 函数 ADC-Regular_Channel_DMA_Cmd.....	64
表 2-24 函数 ADC_High_Priority_Channel_DMA_Cmd.....	65
表 2-25 函数 ADC_Double_Mode_Config.....	65
表 2-26 函数 ADC_Reference_Voltage_Config.....	66
表 2-27 函数 ADC_Analog_Watchdog_Channel_Config.....	66
表 2-28 函数 ADC-Regular_Channel_Watchdog_Enable.....	66
表 2-29 函数 ADC_External_Trig_Conv_Enable.....	67
表 2-30 函数 ADC_External_Trig_Conv_Config.....	67
表 2-31 函数 ADC-Regular_Channel_Config.....	69
表 2-32 函数 ADC-Regular_Sequencer_Length_Config.....	69
表 2-33 函数 ADC-Regular_Channel_TxCCRy_Trig_Enable.....	70
表 2-34 函数 ADC_Software_Start_Conv.....	70
表 2-35 函数 ADC_Continuous_Mode_Cmd.....	71
表 2-36 函数 ADC_Disc_Mode_Channel_Count_Config.....	71
表 2-37 函数 ADC_Disc_Mode_Cmd.....	71
表 2-38 函数 ADC_Get_Conversion_Value.....	72

表 2-39 函数 ADC_High_Priority_Watchdog_Enable.....	72
表 2-40 函数 ADC_HPExternal_Trig_Conv_Enable.....	72
表 2-41 函数 ADC_High_Priority_Channel_Config.....	72
表 2-42 函数 ADC_High_Priority_Sequencer_Length_Config.....	73
表 2-43 函数 ADC_Set_HPChannel_Conv_Value_Offset.....	73
表 2-44 函数 ADC_HPExternal_Trig_Conv_Config.....	74
表 2-45 函数 ADC_Software_HPStart_Conv.....	76
表 2-46 函数 ADC_HPAuto_Conv_Cmd.....	76
表 2-47 函数 ADC_HPDisc_Mode_Cmd.....	76
表 2-48 函数 ADC_Get_HPConversion_Data.....	76
表 2-49 函数 ADC_Set_INT_Enable.....	77
表 2-50 函数 ADC_Get_INT_Flag.....	77
表 2-51 函数 ADC_Clear_INT_Flag.....	78
表 2-52 函数 ADC_Get_INT_Status.....	78
表 3-1 ATIM/ECCP 寄存器结构说明.....	80
表 3-2 ATIM 配置信息结构体说明.....	82
表 3-3 ECCP 捕捉模式配置信息结构体说明.....	82
表 3-4 ECCP 捕捉模式配置信息结构体说明.....	83
表 3-5 ATIM 固件库函数列表.....	83
表 3-6 ECCP 固件库函数列表.....	85
表 3-7 函数 ATIM_X_Configuration.....	87
表 3-8 函数 ATIM_Z_Configuration.....	87
表 3-9 函数 ATIM_Struct_Init.....	87
表 3-10 函数 ATIM_X_Cmd.....	88
表 3-11 函数 ATIM_Z_Cmd.....	88
表 3-12 函数 ATIM_X_Udata_Configuration.....	88
表 3-13 函数 ATIM_Z_Udata_Configuration.....	88
表 3-14 函数 ATIM_X_Udata_Cmd.....	89
表 3-15 函数 ATIM_Z_Udata_Cmd.....	89
表 3-16 函数 ATIM_X_Set_Counter.....	89
表 3-17 函数 ATIM_Z_Set_Counter.....	90
表 3-18 函数 ATIM_X_Set_Period.....	90
表 3-19 函数 ATIM_Z_Set_Period.....	90
表 3-20 函数 ATIM_X_Set_Prescaler.....	91
表 3-21 函数 ATIM_Z_Set_Prescaler.....	91
表 3-22 函数 ATIM_X_Counter_Mode_Config.....	91
表 3-23 函数 ATIM_Z_Counter_Mode_Config.....	92
表 3-24 函数 ATIM_X_Clock_Config.....	92
表 3-25 函数 ATIM_Z_Clock_Config.....	92
表 3-26 函数 ATIM_X_Postscaler_Config.....	93
表 3-27 函数 ATIM_Z_Postscaler_Config.....	93
表 3-28 函数 ATIM_X_External_Pulse_Sync_Config.....	94
表 3-29 函数 ATIM_Z_External_Pulse_Sync_Config.....	94
表 3-30 函数 ATIM_X_Work_Mode_Config.....	95
表 3-31 函数 ATIM_Z_Work_Mode_Config.....	95

表 3-32 函数 ATIM_X_Get_Direction.....	96
表 3-33 函数 ATIM_Z_Get_Direction.....	96
表 3-34 函数 ATIM_X_Overflow_AD_Enable.....	96
表 3-35 函数 ATIM_Z_Overflow_AD_Enable.....	96
表 3-36 函数 ATIM_X_Underflow_AD_Enable.....	97
表 3-37 函数 ATIM_Z_Underflow_AD_Enable.....	97
表 3-38 函数 ATIM_X_TriggerAD_Config.....	97
表 3-39 函数 ATIM_Z_TriggerAD_Config.....	98
表 3-40 函数 ATIM_X_Set_TriggerAD_Signal.....	98
表 3-41 函数 ATIM_Z_Set_TriggerAD_Signal.....	98
表 3-42 函数 ATIM_X_Updata_Immediately_Config.....	99
表 3-43 函数 ATIM_Z_Updata_Immediately_Config.....	99
表 3-44 函数 ATIM_X_Updata_Output_Ctl.....	99
表 3-45 函数 ATIM_Z_Updata_Output_Ctl.....	99
表 3-46 函数 ATIM_X_Updata_Enable.....	100
表 3-47 函数 ATIM_Z_Updata_Enable.....	100
表 3-48 函数 ATIM_X_Set_Updata_Counter.....	100
表 3-49 函数 ATIM_Z_Set_Updata_Counter.....	101
表 3-50 函数 ATIM_X_Slave_Mode_Config.....	101
表 3-51 函数 ATIM_Z_Slave_Mode_Config.....	101
表 3-52 函数 ATIM_Master_Mode_Config.....	102
表 3-53 函数 ATIM_Master_Slave_Snyc_Enable.....	102
表 3-54 函数 ATIM_Trigger_Select_Config.....	103
表 3-55 函数 ATIM_Timer_Unite_Enable.....	103
表 3-56 函数 ATIM_X_Get_Counter.....	103
表 3-57 函数 ATIM_Z_Get_Counter.....	104
表 3-58 函数 ATIM_X_Get_Period.....	104
表 3-59 函数 ATIM_Z_Get_Period.....	104
表 3-60 函数 ATIM_X_Get_Prescaler.....	105
表 3-61 函数 ATIM_Z_Get_Prescaler.....	105
表 3-62 函数 ATIM_X_Updata_INT_Enable.....	105
表 3-63 函数 ATIM_Z_Updata_INT_Enable.....	105
表 3-64 函数 ATIM_X_Overflow_INT_Enable.....	106
表 3-65 函数 ATIM_Z_Overflow_INT_Enable.....	106
表 3-66 函数 ATIM_X_Trigger_INT_Enable.....	106
表 3-67 函数 ATIM_X_Updata_DMA_Enable.....	107
表 3-68 函数 ATIM_Z_Updata_DMA_Enable.....	107
表 3-69 函数 ATIM_X_Trigger_DMA_Enable.....	107
表 3-70 函数 ATIM_X_Get_Updata_INT_Flag.....	108
表 3-71 函数 ATIM_Z_Get_Updata_INT_Flag.....	108
表 3-72 函数 ATIM_X_Get_Overflow_INT_Flag.....	108
表 3-73 函数 ATIM_Z_Get_Overflow_INT_Flag.....	108
表 3-74 函数 ATIM_X_Get_Trigger_INT_Flag.....	109
表 3-75 函数 ATIM_X_Generate_Trigger_Config.....	109
表 3-76 函数 ATIM_X_Get_Updata_DMA_INT_Flag.....	109

表 3-77 函数 ATIM_Z_Get_Updata_DMA_INT_Flag.....	110
表 3-78 函数 ATIM_X_Get_Trigger_DMA_INT_Flag.....	110
表 3-79 函数 ATIM_X_Clear_Updata_INT_Flag.....	110
表 3-80 函数 ATIM_Z_Clear_Updata_INT_Flag.....	110
表 3-81 函数 ATIM_X_Clear_Overflow_INT_Flag.....	111
表 3-82 函数 ATIM_Z_Clear_Overflow_INT_Flag.....	111
表 3-83 函数 ATIM_X_Clear_Trigger_INT_Flag.....	111
表 3-84 函数 ECCP_Compare_Configuration.....	111
表 3-85 函数 ECCP_Capture_Configuration.....	112
表 3-86 函数 ECCP_Capture_Struct_Init.....	112
表 3-87 函数 ECCP_PWM_Configuration.....	113
表 3-88 函数 ECCP_PWM_Struct_Init.....	113
表 3-89 函数 ECCP_Capture_Mode_Config.....	113
表 3-90 函数 ECCP_Compare_Mode_Config.....	114
表 3-91 函数 ECCP_PWM_Mode_Config.....	114
表 3-92 函数 ECCP_Get_Capture_Result.....	115
表 3-93 函数 ECCP_Get_Compare_Result.....	115
表 3-94 函数 ECCP_Set_Compare_Result.....	116
表 3-95 函数 ECCP_Generate_Trigger_Config.....	116
表 3-96 函数 ECCP_PWM_Input_Enable.....	117
表 3-97 函数 ECCP_Input_XOR_Enable.....	117
表 3-98 函数 ECCP_Single_Pulse_Enable.....	117
表 3-99 函数 ECCP_Single_Pulse_Shut_Enable.....	118
表 3-100 函数 ECCP_PWM_Restart_Enable.....	118
表 3-101 函数 ECCP_Dead_Time_Config.....	118
表 3-102 函数 ECCP_Channel_Output_Control.....	119
表 3-103 函数 ECCP_Channel_Output_Mode.....	119
表 3-104 函数 ECCP_Channel_Work_State_Config.....	120
表 3-105 函数 ECCP_Get_Channel_Work_State.....	120
表 3-106 函数 ECCP_CHANNEL4_Shutdown_SEL.....	121
表 3-107 函数 ECCP_Channel_Shutdown_Signal.....	121
表 3-108 函数 ECCP_Channel_Pin_Ctl.....	122
表 3-109 函数 ECCP_Zero_Clock_Config.....	123
表 3-110 函数 ECCP_Channel_Pin_Tristate_Enable.....	123
表 3-111 函数 ECCP_Channel_INT_Enable.....	124
表 3-112 函数 ECCP_X_Turn_off_DMA_Enable.....	124
表 3-113 函数 ECCP_Channel_DMA_Enable.....	125
表 3-114 函数 ECCP_Get_Channel_Trigger_INT_Flag.....	125
表 3-115 函数 ECCP_X_Get_Turn_off_DMA_Flag.....	125
表 3-116 函数 ECCP_Get_Trigger_DMA_INT_Flag.....	126
表 3-117 函数 ECCP_Clear_Channel_INT_Flag.....	126
表 3-118 函数 ECCP_PWM_Move_Phase_Enable.....	127
表 3-119 函数 ECCP_Channel_Zero_Detect_Sequential_Ctl.....	127
表 3-120 函数 ECCP_Get_Channel_Zero_Detection_State.....	127
表 3-121 函数 ECCP_Clear_Channel_Zero_Detection_State.....	128

表 3-122 函数 ECCP_Channel_Zero_Detect_Enable.....	128
表 3-123 函数 ECCP_Channel_Zero_Voltage_Config.....	129
表 4-1 BKP 寄存器结构说明.....	130
表 4-2 BKP 固件库函数列表.....	131
表 4-3 函数 BKP_Reset.....	131
表 4-4 函数 BKP_Write_And_Read_Enable.....	132
表 4-5 函数 BKP_Reset_Enable.....	132
表 4-6 函数 BKP_Pin_Effective_Level_Config.....	132
表 4-7 函数 BKP_Pin_Enable.....	133
表 4-8 函数 BKP_RTC_Clock_Config.....	133
表 4-9 函数 BKP_External_Clock_Bypass_Enable.....	134
表 4-10 函数 BKP_Data_Config.....	134
表 4-11 函数 BKP_Get_Data.....	134
表 4-12 函数 BKP_Get_Data.....	134
表 4-13 函数 BKP_Get_Pin_TAMP_INT_Flag.....	135
表 4-14 函数 BKP_Clear_Pin_TAMP_INT_Flag.....	135
表 5-1 BTIM 寄存器结构说明.....	136
表 5-2 BTIM 固件库函数列表.....	138
表 5-3 函数 TIM_Reset.....	139
表 5-4 函数 BTIM_Configuration.....	139
表 5-5 函数 BTIM_Struct_Init.....	139
表 5-6 函数 BTIM_Cmd.....	140
表 5-7 函数 BTIM_Set_Counter.....	140
表 5-8 函数 BTIM_Set_Period.....	140
表 5-9 函数 BTIM_Set_Prescaler.....	141
表 5-10 函数 BTIM_Counter_Mode_Config.....	141
表 5-11 函数 BTIM_Clock_Config.....	141
表 5-12 函数 BTIM_External_Pulse_Sync_Config.....	142
表 5-13 函数 BTIM_Work_Mode_Config.....	142
表 5-14 函数 BTIM_Generate_Trigger_Config.....	142
表 5-15 函数 BTIM_Single_Pulse_Enable.....	143
表 5-16 函数 BTIM_Single_Pulse_Shut_Enable.....	143
表 5-17 函数 BTIM_Update_Immediately_Config.....	143
表 5-18 函数 BTIM_Master_Slave_Snyc_Config.....	144
表 5-19 函数 BTIM_Trigger_Select_Config.....	144
表 5-20 函数 BTIM_Slave_Mode_Config.....	144
表 5-21 函数 BTIM_Master_Mode_Config.....	145
表 5-22 函数 BTIM_Update_Rising_Edge_Config.....	145
表 5-23 函数 BTIM_Update_Enable.....	146
表 5-24 函数 BTIM_Get_Direction.....	146
表 5-25 函数 BTIM_Get_Counter.....	146
表 5-26 函数 BTIM_Get_Period.....	147
表 5-27 函数 BTIM_Get_Prescaler.....	147
表 5-28 函数 BTIM_Trigger_DMA_Enable.....	147
表 5-29 函数 BTIM_Update_DMA_Enable.....	147

表 5-30 函数 BTIM_Overflow_INT_Enable.....	148
表 5-31 函数 BTIM_Trigger_INT_Enable.....	148
表 5-32 函数 BTIM_Updata_INT_Enable.....	148
表 5-33 函数 BTIM_Updata_Enable.....	149
表 5-34 函数 BTIM_Get_Updata_DMA_INT_Status.....	149
表 5-35 函数 BTIM_Get_Overflow_INT_Status.....	149
表 5-36 函数 BTIM_Get_Trigger_INT_Status.....	149
表 5-37 函数 BTIM_Get_Updata_INT_Status.....	150
表 5-38 函数 BTIM_Get_Trigger_DMA_INT_Flag.....	150
表 5-39 函数 BTIM_Get_Updata_DMA_INT_Flag.....	150
表 5-40 函数 BTIM_Get_Overflow_INT_Flag.....	150
表 5-41 函数 BTIM_Get_Updata_INT_Flag.....	151
表 5-42 函数 BTIM_Clear_Overflow_INT_Flag.....	151
表 5-43 函数 BTIM_Clear_Trigger_INT_Flag.....	151
表 5-44 函数 BTIM_Clear_Updata_INT_Flag.....	151
表 6-1 CAN_InitTypeDef 寄存器结构说明.....	152
表 6-2 CAN_ErrorTypeDef 寄存器结构说明.....	153
表 6-3 CAN_MessageTypeDef 寄存器结构说明.....	153
表 6-4 CAN 固件库函数列表.....	154
表 6-5 函数 CAN_Reset.....	155
表 6-6 函数 CAN_Configuration.....	155
表 6-7 函数 CAN_Struct_Init.....	156
表 6-8 函数 CAN_Get_Receive_Message_Counter.....	156
表 6-9 函数 CAN_Get_Transmit_Status.....	156
表 6-10 函数 CAN_Cmd.....	156
表 6-11 函数 CAN_Clock_Source_Config.....	157
表 6-12 函数 CAN_Sleep_Mode_Enable.....	157
表 6-13 函数 CAN_Reset_Mode_Enable.....	157
表 6-14 函数 CAN_Work_Mode_Config.....	158
表 6-15 函数 CAN_Bus_Sample_Times_Config.....	158
表 6-16 函数 CAN_Time_Segment_Config.....	158
表 6-17 函数 CAN_Sync_Jump_Width_Config.....	159
表 6-18 函数 CAN_Baud_Rate_Preset_Config.....	159
表 6-19 函数 CAN_Get_Error_Code.....	159
表 6-20 函数 CAN_Get_Error_Warning_Limit.....	160
表 6-21 函数 CAN_Get_Error_Counter.....	160
表 6-22 函数 CAN_Error_Warning_Limit_Config.....	160
表 6-23 函数 CAN_Error_Counter_Config.....	161
表 6-24 函数 CAN_Acceptance_Config.....	161
表 6-25 函数 CAN_Acceptance_Config.....	161
表 6-26 函数 CAN_Acceptance_Mask_Config.....	161
表 6-27 函数 CAN_Get_Acceptance_Mask.....	162
表 6-28 函数 CAN_Transmit_Message_Configuration.....	162
表 6-29 函数 CAN_Receive_Message_Configuration.....	162
表 6-30 函数 CAN_Message_Struct_Init.....	163

表 6-31 函数 CAN_Clear_Buffer_Overflow_Flag.....	163
表 6-32 函数 CAN_Release_Receive_Buffer.....	163
表 6-33 函数 CAN_Transmit_Single.....	163
表 6-34 函数 CAN_Transmit_Repeat.....	164
表 6-35 函数 CAN_Frame_Format_Config.....	164
表 6-36 函数 CAN_Remote_Request_Config.....	164
表 6-37 函数 CAN_Data_Length_Config.....	165
表 6-38 函数 CAN_Identification_Code_Config.....	165
表 6-39 函数 CAN_Get_INT_Flag.....	165
表 6-40 函数 CAN_Clear_INT_Flag.....	166
表 6-41 函数 CAN_Set_INT_Enable.....	166
表 7-1CMP 寄存器结构说明.....	168
表 7-2 函数 CMP_Reset.....	171
表 7-3 函数 CMP_Configuration.....	171
表 7-4 函数 CMP_Struct_Init.....	171
表 7-5 函数 CMP0_Cmd.....	172
表 7-6 函数 CMP1_Cmd.....	172
表 7-7 函数 CMP2_Cmd.....	172
表 7-8 函数 CMP3_Cmd.....	172
表 7-9 函数 CMP0_POSITIVE_INPUT_SELECT.....	173
表 7-10 函数 CMP0_NEGATIVE_INPUT_SELECT.....	173
表 7-11 函数 CMP1_POSITIVE_INPUT_SELECT.....	173
表 7-12 函数 CMP1_NEGATIVE_INPUT_SELECT.....	174
表 7-13 函数 CMP2_POSITIVE_INPUT_SELECT.....	174
表 7-14 函数 CMP2_NEGATIVE_INPUT_SELECT.....	175
表 7-15 函数 CMP3_POSITIVE_INPUT_SELECT.....	175
表 7-16 函数 CMP3_NEGATIVE_INPUT_SELECT.....	176
表 7-17 函数 CMP0_OUTPUT_POL_SELECT.....	176
表 7-18 函数 CMP1_OUTPUT_POL_SELECT.....	176
表 7-19 函数 CMP2_OUTPUT_POL_SELECT.....	177
表 7-20 函数 CMP3_OUTPUT_POL_SELECT.....	177
表 7-21 函数 CMP_OUTPUT_SELECT.....	177
表 7-22 函数 CMP0_Read_Output_State.....	177
表 7-23 函数 CMP1_Read_Output_State.....	178
表 7-24 函数 CMP2_Read_Output_State.....	178
表 7-25 函数 CMP3_Read_Output_State.....	178
表 7-26 函数 CMP0_Get_Updata_INT_Flag.....	178
表 7-27 函数 CMP1_Get_Updata_INT_Flag.....	179
表 7-28 函数 CMP2_Get_Updata_INT_Flag.....	179
表 7-29 函数 CMP3_Get_Updata_INT_Flag.....	179
表 7-30 函数 CMP_Trigger_Select_Config.....	179
表 7-31 函数 CMP0_Clear_Trigger_INT_Flag.....	180
表 7-32 函数 CMP1_Clear_Trigger_INT_Flag.....	180
表 7-33 函数 CMP2_Clear_Trigger_INT_Flag.....	180
表 7-34 函数 CMP3_Clear_Trigger_INT_Flag.....	180

表 7-35 函数 CMP0_INT_Enable.....	180
表 7-36 函数 CMP1_INT_Enable.....	181
表 7-37 函数 CMP2_INT_Enable.....	181
表 7-38 函数 CMP3_INT_Enable.....	181
表 7-39 函数 CMP_SluggishVoltage_Select.....	182
表 7-40 函数 CMP_HALLMODE_Select.....	182
表 7-41 函数 CMP_BEMF_Enable.....	182
表 7-42 函数 CMP_FLTINSEL_Select.....	182
表 8-1 CRC 寄存器结构说明.....	184
表 8-2 CRC 配置信息结构体说明.....	185
表 8-3 CRC 固件库函数列表.....	186
表 8-4 函数 CRC_Reset.....	186
表 8-5 函数 CRC_Configuration.....	186
表 8-6 函数 CRC_Struct_Init.....	187
表 8-7 函数 CRC_INPUT_DATA.....	187
表 8-8 函数 CRC_GET_RESULT.....	187
表 8-9 函数 CRC_SET_INITVALUE.....	187
表 8-10 函数 CRC_SET_PLN.....	188
表 8-11 函数 CRC_SET_RXOR.....	188
表 8-12 函数 CRC_SET_IDATA.....	188
表 8-13 函数 CRC_GET_TEMP.....	188
表 8-14 函数 CRC_SET_RSET.....	189
表 9-1 DMA 寄存器结构说明.....	192
表 9-2 DMA 配置信息结构体说明.....	194
表 9-3 DMA 固件库函数列表.....	194
表 9-4 函数 DMA_Reset.....	195
表 9-5 函数 DMA_Configuration.....	196
表 9-6 函数 DMA_Struct_Init.....	196
表 9-7 函数 DMA_Transfer_Number_Config.....	196
表 9-8 函数 DMA_Memory_To_Memory_Enable.....	197
表 9-9 函数 DMA_Channel_Priority_Config.....	197
表 9-10 函数 DMA_Peripheral_Data_Width_Config.....	198
表 9-11 函数 DMA_Memory_Data_Width_Config.....	198
表 9-12 函数 DMA_Peripheral_addr_increase_Enable.....	199
表 9-13 函数 DMA_Memory_addr_increase_Enable.....	199
表 9-14 函数 DMA_Loop_Mode_Enable.....	200
表 9-15 函数 DMA_Transfer_Direction_Config.....	200
表 9-16 函数 DMA_Transfer_Mode_Config.....	201
表 9-17 函数 DMA_Transfer_Mode_Config.....	202
表 9-18 函数 DMA_Channel_Enable.....	202
表 9-19 函数 DMA_Peripheral_Start_Address_Config.....	203
表 9-20 函数 DMA_Transfer_Mode_Config.....	203
表 9-21 函数 DMA_Get_Peripheral_Current_Address.....	204
表 9-22 函数 DMA_Get_Memory_Current_Address.....	204
表 9-23 函数 DMA_Get_Transfer_Number_Remain.....	205

表 9-24 函数 DMA_Get_INT_Flag.....	205
表 9-25 函数 DMA_Clear_INT_Flag.....	206
表 9-26 函数 DMA_Set_INT_Enable.....	206
表 9-27 函数 DMA_Get_Error_Transfer_INT_Flag.....	207
表 9-28 函数 DMA_Get_Half_Transfer_INT_Flag.....	207
表 9-29 函数 DMA_Get_Finish_Transfer_INT_Flag.....	208
表 9-30 函数 DMA_Error_Transfer_INT_Enable.....	208
表 9-31 函数 DMA_Half_Transfer_INT_Enable.....	209
表 9-32 函数 DMA_Finish_Transfer_INT_Enable.....	209
表 10-1 FLASH 寄存器结构说明.....	212
表 10-2 FLASH 锁定解锁状态.....	212
表 10-3 FLASH CheckSum 结果数据结构体说明.....	213
表 10-4 FLASH 编程信息结构体说明.....	213
表 10-5 FLASH 固件库函数列表.....	213
表 10-6 函数 CHECK_RESTRICTION_RAM.....	215
表 10-7 函数 SFR_Config_RAM.....	216
表 10-8 函数 FLASH_Get_NonVolatile_Memory_Unlock_Status_RAM.....	216
表 10-9 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM.....	216
表 10-10 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status.....	217
表 10-11 函数 FLASH_Unlock_ISP_RAM.....	217
表 10-12 函数 FLASH_Unlock_ISP.....	217
表 10-13 函数 FLASH_Get_Flash_Unlock_Status_RAM.....	217
表 10-14 函数 FLASH_Clear_Flash_Unlock_Status_RAM.....	218
表 10-15 函数 FLASH_Clear_Flash_Unlock_Status.....	218
表 10-16 函数 FLASH_Unlock_Code_RAM.....	218
表 10-17 函数 FLASH_Unlock_Code.....	219
表 10-18 函数 FLASH_Get_Config_Unlock_Status_RAM.....	219
表 10-19 函数 FLASH_Clear_Config_Unlock_Status_RAM.....	219
表 10-20 函数 FLASH_Clear_Config_Unlock_Status.....	219
表 10-21 函数 FLASH_Unlock_User_Config_RAM.....	220
表 10-22 函数 FLASH_Unlock_User_Config.....	220
表 10-23 函数 FLASH_Data_Write_Enable_RAM.....	220
表 10-24 函数 FLASH_Zone_Config_RAM.....	221
表 10-25 函数 FLASH_Zone_Config.....	221
表 10-26 函数 FLASH_Standby_Mode_Config_RAM.....	221
表 10-27 函数 FLASH_Read_Mode_Config_RAM.....	222
表 10-28 函数 FLASH_Calibration_Update_Enable_RAM.....	222
表 10-29 函数 FLASH_Information_Zone_Wipe_Unlock_Config_RAM.....	222
表 10-30 函数 FLASH_Half_Page_Write_Size_Config_RAM.....	223
表 10-31 函数 FLASH_Program_Cmd_Config_RAM.....	223
表 10-32 函数 FLASH_Executive_Cmd_RAM.....	223
表 10-33 函数 FLASH_Executive_Cmd.....	224
表 10-34 函数 FLASH_NonVolatile_Memory_ECC_Enable_RAM.....	224
表 10-35 函数 FLASH_Linear_Prefetch_Enable_RAM.....	224
表 10-36 函数 FLASH_Period_Number_Config_RAM.....	224

表 10-37 函数 FLASH_Program_Addr_Config_RAM.....	225
表 10-38 函数 FLASH_Get_Program_Status_RAM.....	225
表 10-39 函数 FLASH_Get_Program_Status.....	225
表 10-40 函数 FLASH_Get_Wipe_Status_RAM.....	226
表 10-41 函数 FLASH_Get_Wipe_Status.....	226
表 10-42 函数 FLASH_Get_Compute_Complete_Status_RAM.....	226
表 10-43 函数 FLASH_Clear_Compute_Complete_Status_RAM.....	226
表 10-44 函数 FLASH_Get_CFG_Error_Flag_RAM.....	227
表 10-45 函数 FLASH_Clear_CFG_Error_Flag_RAM.....	227
表 10-46 函数 FLASH_CheckSum_Addr_Config_RAM.....	227
表 10-47 函数 FLASH_Start_SIG_Compute_Enable_RAM.....	228
表 10-48 函数 FLASH_Get_CheckSum_Result_RAM.....	228
表 10-49 函数 FLASH_Wipe_Configuration_RAM.....	228
表 10-50 函数 FLASH_Wipe_Configuration.....	229
表 10-51 函数 FLASH_Program_Configuration_RAM.....	229
表 10-52 函数 FLASH_Program_Configuration.....	230
表 10-53 函数 Read_Flash_or_CFR_RAM.....	230
表 10-54 函数 Read_Flash_or_CFR.....	231
表 10-55 函数 Read_Soft_Device_ID1.....	231
表 10-56 函数 Read_Soft_Device_ID2.....	232
表 10-57 函数 Read_Soft_Device_ID3.....	232
表 10-58 函数 Read_Soft_Device_ID4.....	232
表 11-1 GPIO 寄存器结构说明.....	234
表 11-2 GPIO 配置信息结构体说明.....	234
表 11-3 GPIO 固件库函数列表.....	235
表 11-4 函数 GPIO_Reset.....	235
表 11-5 函数 GPIO_Configuration.....	235
表 11-6 函数 GPIO_Struct_Init.....	236
表 11-7 函数 GPIO_Pin_Lock_Config.....	236
表 11-8 函数 GPIO_Pull_Up_Enable.....	236
表 11-9 函数 GPIO_Toggle_Pull_Up_Config.....	237
表 11-10 函数 GPIO_Pull_Down_Enable.....	237
表 11-11 函数 GPIO_Toggle_Pull_Down_Config.....	237
表 11-12 函数 GPIO_Open_Drain_Enable.....	238
表 11-13 函数 GPIO_Toggle_Open_Drain_Config.....	238
表 11-14 函数 GPIO_Write_Mode_Bits.....	238
表 11-15 函数 GPIO_Speed_Config.....	239
表 11-16 函数 GPIO_Toggle_Speed_Config.....	239
表 11-17 函数 GPIO_Read_Input_Data_Bit.....	240
表 11-18 函数 GPIO_Read_Input_Data.....	240
表 11-19 函数 GPIO_Read_Output_Data_Bit.....	240
表 11-20 函数 GPIO_Read_Output_Data.....	240
表 11-21 函数 GPIO_Set_Output_Data_Bits.....	241
表 11-22 函数 GPIO_Toggle_Output_Data_Config.....	241
表 11-23 函数 GPIO_Pin_RMP_Config.....	241

表 12-1 GPTIM 寄存器结构说明.....	245
表 12-2 GPTIM 配置信息结构体说明.....	246
表 12-3 GPTIM 固件库函数列表.....	246
表 12-4 函数 GPTIM_Configuration.....	247
表 12-5 函数 GPTIM_Struct_Init.....	248
表 12-6 函数 GPTIM_Cmd.....	248
表 12-7 函数 GPTIM_Set_Counter.....	248
表 12-8 函数 GPTIM_Set_Period.....	249
表 12-9 函数 GPTIM_Set_Prescaler.....	249
表 12-10 函数 GPTIM_Counter_Mode_Config.....	249
表 12-11 函数 GPTIM_Clock_Config.....	250
表 12-12 函数 GPTIM_External_Pulse_Sync_Config.....	250
表 12-13 函数 GPTIM_Work_Mode_Config.....	251
表 12-14 函数 GPTIM_Update_Immediately_Config.....	251
表 12-15 函数 GPTIM_Master_Slave_Snyc_Config.....	251
表 12-16 函数 GPTIM_Trigger_Select_Config.....	252
表 12-17 函数 GPTIM_Slave_Mode_Config.....	252
表 12-18 函数 GPTIM_Master_Mode_Config.....	253
表 12-19 函数 GPTIM_Update_Rising_Edge_Config.....	253
表 12-20 函数 GPTIM_Update_Enable.....	254
表 12-21 函数 GPTIM_Trigger_DMA_Enable.....	254
表 12-22 函数 GPTIM_Update_DMA_Enable.....	255
表 12-23 函数 GPTIM_Update_INT_Enable.....	255
表 12-24 函数 GPTIM_Trigger_INT_Enable.....	255
表 12-25 函数 GPTIM_Generate_Trigger_Config.....	256
表 12-26 函数 GPTIM_Get_Direction.....	256
表 12-27 函数 GPTIM_Get_Counter.....	257
表 12-28 函数 GPTIM_Get_Period.....	257
表 12-29 函数 GPTIM_Get_Prescaler.....	257
表 12-30 函数 GPTIM_Overflow_INT_Enable.....	258
表 12-31 函数 GPTIM_Clear_Overflow_INT_Flag.....	258
表 12-32 函数 GPTIM_Clear_Update_INT_Flag.....	258
表 12-33 函数 GPTIM_Clear_Trigger_INT_Flag.....	259
表 12-34 函数 GPTIM_Get_Overflow_INT_Flag.....	259
表 12-35 函数 GPTIM_Get_Update_INT_Flag.....	259
表 12-36 函数 GPTIM_Get_Trigger_INT_Flag.....	260
表 12-37 函数 GPTIM_Get_Update_DMA_INT_Flag.....	260
表 12-38 函数 GPTIM_Get_Trigger_DMA_INT_Flag.....	260
表 13-1 I2C 寄存器结构说明.....	262
表 13-2 I2C 配置信息结构体说明.....	263
表 13-3 I2C 固件库函数列表.....	264
表 13-4 函数 I2C_Reset.....	265
表 13-5 函数 I2C_Configuration.....	266
表 13-6 函数 I2C_Struct_Init.....	266
表 13-7 函数 I2C_Cmd.....	266

表 13-8 函数 I2C_Bufr_Address_Config.....	267
表 13-9 函数 I2C_Generate_START.....	267
表 13-10 函数 I2C_Generate_STOP.....	267
表 13-11 函数 I2C_Ack_Config.....	267
表 13-12 函数 I2C_Ack_DATA_Config.....	268
表 13-13 函数 I2C_Call_Cmd.....	268
表 13-14 函数 I2C_Clock_Config.....	268
表 13-15 函数 I2C_MATCH_ADDRESS_Config.....	269
表 13-16 函数 I2C_SCL_Enable.....	269
表 13-17 函数 I2C_NMENA_Enable.....	269
表 13-18 函数 I2C_SMBUS_Enable.....	270
表 13-19 函数 I2C_SMBT_Config.....	270
表 13-20 函数 I2C_SMBus_ALERT_Config.....	270
表 13-21 函数 I2C_SendData.....	271
表 13-22 函数 I2C_SendData8.....	271
表 13-23 函数 I2C_ReceiveData.....	271
表 13-24 函数 I2C_ARP_Enable.....	271
表 13-25 函数 I2C_ADDR_Config.....	272
表 13-26 函数 I2C_MSK_Config.....	272
表 13-27 函数 I2C_BRGH_Config.....	272
表 13-28 函数 I2C_BRGL_Config.....	273
表 13-29 函数 I2C_Start_INT_Enable.....	273
表 13-30 函数 I2C_Stop_INT_Enable.....	273
表 13-31 函数 I2C_Ack_Fail_INT_Enable.....	274
表 13-32 函数 I2C_Arbitration_Lost_INT_Enable.....	274
表 13-33 函数 I2C_SMBus_Alert_INT_Enable.....	274
表 13-34 函数 I2C_SMBus_HostHead_INT_Enable.....	274
表 13-35 函数 I2C_SMBus_Device_Defaultaddress_INT_Enable.....	275
表 13-36 函数 I2C_ISIE_INT_Enable.....	275
表 13-37 函数 I2C_Receive_DMA_INT_Enable.....	275
表 13-38 函数 I2C_Transmit_DMA_INT_Enable.....	276
表 13-39 函数 I2C_Get_Start_Flag.....	276
表 13-40 函数 I2C_Clear_Start_Flag.....	276
表 13-41 函数 I2C_Get_Stop_Flag.....	277
表 13-42 函数 I2C_Clear_Stop_Flag.....	277
表 13-43 函数 I2C_Get_Address_Match_Flag.....	277
表 13-44 函数 I2C_Get_HighAddress_Flag.....	277
表 13-45 函数 I2C_Get_Data_Flag.....	278
表 13-46 函数 I2C_Get_Ack_Fail_Flag.....	278
表 13-47 函数 I2C_Clear_Ack_Fail_Flag.....	278
表 13-48 函数 I2C_Get_Arbitration_Lost_Flag.....	279
表 13-49 函数 I2C_Get_Write_Read_Flag.....	279
表 13-50 函数 I2C_Get_SMBus_Alert_Flag.....	279
表 13-51 函数 I2C_Clear_SMBus_Alert_Flag.....	280
表 13-52 函数 I2C_Get_SMBus_Host_Header_Flag.....	280

表 13-53 函数 I2C_Clear_SMBus_Host_Header_Flag.....	280
表 13-54 函数 I2C_Get_SMBus_Device_Default_Flag.....	280
表 13-55 函数 I2C_Clear_SMBus_Device_Default_Flag.....	281
表 13-56 函数 I2C_Get_INTERRUPT_Flag.....	281
表 13-57 函数 I2C_Clear_INTERRUPT_Flag.....	281
表 13-58 函数 I2C_Get_Receive_Buff_Flag.....	282
表 13-59 函数 I2C_Get_Transmit_Buff_Flag.....	282
表 13-60 函数 I2C_Get_Receive_DMA_Flag.....	282
表 13-61 函数 I2C_Get_Transmit_DMA_Flag.....	282
表 14-1 INT 寄存器结构说明.....	285
表 14-2 INT 配置信息结构体说明.....	286
表 14-3 INT 固件库函数列表.....	287
表 14-4 函数 INT_Get_Interrupt_Action.....	288
表 14-5 函数 INT_Get_Priority_Pending_Action.....	288
表 14-6 函数 INT_Priority_Base.....	288
表 14-7 函数 INT_Get_Priority_Base.....	289
表 14-8 函数 INT_Stack_Align_Config.....	289
表 14-9 函数 INT_Fault_Masking_Config.....	289
表 14-10 函数 INT_Get_Pre_Empty.....	289
表 14-11 函数 INT_Get_Pending_Flag.....	290
表 14-12 函数 INT_Priority_Group_Config.....	290
表 14-13 函数 INT_Get_Priority_Group.....	290
表 14-14 函数 INT_All_Enable.....	291
表 14-15 函数 INT_Interrupt_Enable.....	291
表 14-16 函数 INT_Set_Systick_Flag.....	291
表 14-17 函数 INT_Set_PendSV_Flag.....	291
表 14-18 函数 INT_Get_Interrupt_Flag.....	292
表 14-19 函数 INT_Clear_Interrupt_Flag.....	292
表 14-20 函数 INT_Interrupt_Priority_Config.....	292
表 14-21 函数 INT_Set_Interrupt_Priority.....	293
表 14-22 函数 INT_Stack_Delay_Enable.....	293
表 14-23 函数 INT_External_Configuration.....	293
表 14-24 函数 INT_External_Struct_Init.....	294
表 14-25 函数 INT_External_Mask_Enable.....	294
表 14-26 函数 INT_External_Rise_Enable.....	294
表 14-27 函数 INT_External_Fall_Enable.....	295
表 14-28 函数 INT_Get_External_Flag.....	295
表 14-29 函数 INT_External_Clear_Flag.....	295
表 14-30 函数 INT_External_Source_Enable.....	296
表 15-1 IWDG 寄存器结构说明.....	297
表 15-2 IWDG 固件库函数列表.....	298
表 15-3 函数 IWDG_Prescaler_Config.....	298
表 15-4 函数 IWDG_Overflow_Config.....	298
表 15-5 函数 IWDG_Overflow_Config.....	299
表 15-6 函数 WDT_Feed_The_Dog.....	299

表 16-1 OP 寄存器结构说明.....	300
表 16-2 函数 OP_Reset.....	301
表 16-3 函数 OP_CAL_Configure.....	301
表 16-4 函数 OP_GAIN_SELSECT.....	302
表 16-5 函数 OP3_POSITIVE_INPUT_SELSECT.....	302
表 16-6 函数 OP_OUTPUT_EN.....	302
表 16-7 函数 OP_MODULE_EN.....	302
表 17-1 OSC 寄存器结构说明.....	304
表 17-2 PLL 寄存器结构说明.....	305
表 17-3 OSC 固件库函数列表.....	305
表 17-4 函数 OSC_SCLK_Configuration.....	307
表 17-5 函数 OSC_HFCK_ConfigurationOSC_SCLK_Configuration.....	307
表 17-6 函数 OSC_LFCK_ConfigurationOSC_SCLK_Configuration.....	307
表 17-7 函数 OSC_CK48M_Configuration.....	308
表 17-8 函数 OSC_Struct_Init.....	308
表 17-9 函数 OSC_LFCK_Division_Config.....	308
表 17-10 函数 OSC_HFCK_Division_Config.....	309
表 17-11 函数 OSC_SCK_Division_Config.....	309
表 17-12 函数 OSC_PLL_Input_Source_Config.....	310
表 17-13 函数 OSC_HFCK_Source_Config.....	310
表 17-14 函数 OSC_HFCK_Enable.....	310
表 17-15 函数 OSC_HFCK_Enable.....	311
表 17-16 函数 OSC_LFCK_Enable.....	311
表 17-17 函数 OSC_SCK_Source_Config.....	311
表 17-18 函数 OSC_Backup_Write_Read_Enable.....	312
表 17-19 函数 OSC_SCLK_Output_Enable.....	312
表 17-20 函数 OSC_SCLK_Output_Select.....	312
表 17-21 函数 OSC_SCLK_Output_Division_Config.....	313
表 17-22 函数 OSC_Clock_Failure_Check_Enable.....	313
表 17-23 函数 OSC_CK48M_Division_Config.....	313
表 17-24 函数 OSC_CK48M_Source_Config.....	314
表 17-25 函数 OSC_CK48M_Enable.....	314
表 17-26 函数 OSC_PLL_Multiple_Value_Select.....	314
表 17-27 函数 OSC_PLL_RST.....	315
表 17-28 函数 OSC_PLL_Start_Delay_Config.....	315
表 17-29 函数 OSC_EXTHF_Start_Delay_Config.....	315
表 17-30 函数 OSC_EXTLF_Start_Delay_Config.....	316
表 17-31 函数 OSC_PLL_Software_Enable.....	316
表 17-32 函数 OSC_EXTHF_Software_Enable.....	317
表 17-33 函数 OSC_EXTLF_Software_Enable.....	317
表 17-34 函数 OSC_INTHF_Software_Enable.....	317
表 17-35 函数 OSC_INTLF_Software_Enable.....	317
表 17-36 函数 OSC_Zero_Drift_Config.....	318
表 17-37 函数 OSC_Positive_Drift_Config.....	318
表 17-38 函数 OSC_Negative_Drift_Config.....	319

表 17-39 函数 OSC_Current_Gain_Config.....	319
表 17-40 函数 OSC_High_Speed_Enable.....	319
表 17-41 函数 OSC_External_Input_Enable.....	319
表 17-42 函数 OSC_Feedback_Resistance_Config.....	320
表 17-43 函数 OSC_PLL_Zero_Source_Enable.....	320
表 17-44 函数 OSC_PLL_Vref2_Enable.....	320
表 17-45 函数 OSC_PLL_Vref0p5_Enable.....	321
表 17-46 函数 OSC_PLL_Vref1p2_Enable.....	321
表 17-47 函数 OSC_PLL_Low_Power_20nA_Enable.....	321
表 17-48 函数 OSC_PLL_Vref1p14_Enable.....	321
表 17-49 函数 OSC_PLL_Low_Power_100nA_Enable.....	322
表 17-50 函数 OSC_PLL_LDO_Enable.....	322
表 17-51 函数 OSC_PLL_INT_Enable.....	322
表 17-52 函数 OSC_EXTHF_INT_Enable.....	323
表 17-53 函数 OSC_EXTLF_INT_Enable.....	323
表 17-54 函数 OSC_INTHF_INT_Enable.....	323
表 17-55 函数 OSC_INTLF_INT_Enable.....	323
表 17-56 函数 OSC_Get_Clock_Failure_INT_Flag.....	324
表 17-57 函数 OSC_Get_PLL_INT_Flag.....	324
表 17-58 函数 OSC_Get_EXTHF_INT_Flag.....	324
表 17-59 函数 OSC_Get_EXTLF_INT_Flag.....	324
表 17-60 函数 OSC_Get_INTHF_INT_Flag.....	325
表 17-61 函数 OSC_Get_INTLF_INT_Flag.....	325
表 17-62 函数 OSC_Get_LP4MIF_INT_Flag.....	325
表 18-1 PCLK 寄存器结构说明.....	326
表 18-2 PCLK 固件库函数列表.....	327
表 18-3 函数 PCLK_CTL0_Peripheral_Clock_Enable.....	327
表 18-4 函数 PCLK_CTL1_Peripheral_Clock_Enable.....	327
表 18-5 函数 PCLK_CTL2_Peripheral_Clock_Enable.....	328
表 18-6 函数 PCLK_CTL3_Peripheral_Clock_Enable.....	329
表 19-1 PM 寄存器结构说明.....	331
表 19-2 PM 固件库函数列表.....	332
表 19-3 函数 PM_IO_Latch_Enable.....	334
表 19-4 函数 PM_Get_IO_Latch_Status.....	334
表 19-5 函数 PM_Internal_Low_Frequency_Enable.....	335
表 19-6 函数 PM_External_Low_Frequency_Enable.....	335
表 19-7 函数 PM_External_Low_Frequency_Clock_Enable.....	335
表 19-8 函数 PM_Main_Bandgap_Enable.....	336
表 19-9 函数 PM_LDO18_Enable.....	336
表 19-10 函数 PM_Backup_Registers_Reset_Config.....	336
表 19-11 函数 PM_Independent_Watchdog_Reset_Config.....	336
表 19-12 函数 PM_SRAMA_In_Standby_Work_Mode_Config.....	337
表 19-13 函数 PM_LPRAM_In_Standby_Work_Mode_Config.....	337
表 19-14 函数 PM_Backup_POR_Delay_Time_Config.....	337
表 19-15 函数 PM_Main_POR_Delay_Time_Config.....	338

表 19-16 函数 PM_Peripheral_IO_Port_Config.....	338
表 19-17 函数 PM_OCAL0LOCK_Enable.....	338
表 19-18 函数 PM_MEMSEL_Enable.....	339
表 19-19 函数 PM_Flash_Power_Off_Enable.....	339
表 19-20 函数 PM_CCP0CLKLPEN_Enable.....	339
表 19-21 函数 PM_Backup_Write_And_Read_Enable.....	339
表 19-22 函数 PM_VREF_Software_Enable.....	340
表 19-23 函数 PM_VREF_SELECT.....	340
表 19-24 函数 PM_LPR_Software_Enable.....	340
表 19-25 函数 PM_Low_Power_Mode_Config.....	340
表 19-26 函数 PM_BOR_Enable.....	341
表 19-27 函数 PM_Low_Power_BOR_Enable.....	341
表 19-28 函数 PM_Temperature_Sensor_Enable.....	341
表 19-29 函数 PM_Temperature_Sensor_Buffer_Enable.....	342
表 19-30 函数 PM_Reference_Voltage_Enable.....	342
表 19-31 函数 PM_Internal_Test_Buffer_Clock_Enable.....	342
表 19-32 函数 PM_Internal_Test_Buffer_Clock_Scaler_Config.....	342
表 19-33 函数 PM_PLL0_Output_Buffer_Enable.....	343
表 19-34 函数 PM_PLL1_Output_Buffer_Enable.....	343
表 19-35 函数 PM_PLL2_Output_Buffer_Enable.....	343
表 19-36 函数 PM_PLL0LDO_Output_Buffer_Enable.....	344
表 19-37 函数 PM_PLL1LDO_Output_Buffer_Enable.....	344
表 19-38 函数 PM_PLL2LDO_Output_Buffer_Enable.....	344
表 19-39 函数 PM_Battery_BOR_Config.....	344
表 19-40 函数 PM_Battery_BOR_Enable.....	345
表 19-41 函数 PM_Peripheral_Voltage_Monitoring_Enable.....	345
表 19-42 函数 PM_Voltage_Detection_Config.....	345
表 19-43 函数 PM_Voltage_Detection_Enable.....	346
表 19-44 函数 PM_External_Wakeup_Pin_Enable.....	346
表 19-45 函数 PM_External_Wakeup_Edge_Config.....	346
表 19-46 函数 PM_Stop_Mode_Peripheral_INLF_Enable.....	347
表 19-47 函数 PM_Peripheral_Reset_Config.....	347
表 19-48 函数 PM_Vdd_Por_Enable.....	348
表 19-49 函数 PM_Low_Power_Bandgap_Enable.....	348
表 19-50 函数 PM_Power_Dissipation_Mode_Config.....	348
表 19-51 函数 PM_Power_Dissipation_Mode_Delay_Config.....	348
表 19-52 函数 PM_Internal_Test_Buffer_Enable.....	349
表 19-53 函数 PM_Clear_Reset_And_Wakeup_Flag.....	349
表 19-54 函数 PM_Get_IWDT_Reset_Flag.....	349
表 19-55 函数 PM_Clear_External_Wakeup_Pin_Flag.....	350
表 19-56 函数 PM_Get_Low_Power_Running_State.....	350
表 19-57 函数 PM_Get_LPR_Status.....	351
表 19-58 函数 PM_Get_Peripheral_Voltage_Detection_Status.....	351
表 19-59 函数 PM_Zero_Drift_Current_Config.....	351
表 19-60 函数 PM_BOR_Voltage_Config.....	352

表 19-61 函数 PM_Main_Regulator_Voltage_Config.....	352
表 19-62 函数 PM_Main_Regulator_HV_Enable.....	352
表 19-63 函数 PM_Reference_Calibration_Config.....	353
表 19-64 函数 PM_INTLF_Bias_Current_Config.....	353
表 19-65 函数 PM_EXTLF_Bias_Current_Config.....	354
表 19-66 函数 PM_INTLF_Capacitance_Calibration_Config.....	354
表 19-67 函数 PM_LP_Bias_Calibration_Config.....	354
表 19-68 函数 PM_LPBGP_Pump_Calibration_Config.....	355
表 19-69 函数 PM_EXTLF_N_Bias_Current_Config.....	355
表 19-70 函数 PM_EXTLF_PIN_Selection_Config.....	355
表 19-71 函数 PM_EXTHF_PIN_Selection_Config.....	356
表 19-72 函数 PM_LDO18_Module_Config.....	356
表 19-73 函数 PM_Main_Regulator_Bandgap_Config.....	356
表 19-74 函数 PM_LPR_Module_Config.....	356
表 20-1 SYSTICK 寄存器结构说明.....	358
表 20-2 RTC 配置信息结构体说明.....	359
表 20-3 RTC 配置信息结构体说明.....	359
表 20-4 RTC 配置信息结构体说明.....	360
表 20-5 RTC 固件库函数列表.....	360
表 20-6 函数 RTC_Reset.....	363
表 20-7 函数 RTC_Configuration.....	363
表 20-8 函数 RTC_Time_Struct_Init.....	364
表 20-9 函数 RTC_Date_Struct_Init.....	364
表 20-10 函数 RTC_Struct_Init.....	364
表 20-11 函数 RTC_Get_Time_Configuration.....	364
表 20-12 函数 RTC_Get_Date_Configuration RTC_Alarm_Configuration.....	365
表 20-13 函数 RTC_Alarm_Configuration RTC_Alarm_Struct_Init.....	365
表 20-14 函数 RTC_Alarm_Struct_Init RTC_Clock_Calibration_Config.....	365
表 20-15 函数 RTC_Clock_Calibration_Config.....	366
表 20-16 函数 RTC_Time_Tick_Output_Enable.....	366
表 20-17 函数 RTC_Time_Stamp_Edge_Config.....	366
表 20-18 函数 RTC_Time_Stamp_Edge_Enable.....	367
表 20-19 函数 RTC_Add_One_Hour_Enable.....	367
表 20-20 函数 RTC_Sub_One_Hour_Enable.....	367
表 20-21 函数 RTC_Time_Tick_Config.....	367
表 20-22 函数 RTC_Start_Config.....	368
表 20-23 函数 RTC_Reset_Config.....	368
表 20-24 函数 RTC_Get_Leap_Year_Flag.....	368
表 20-25 函数 RTC_Hour_Format_Config.....	369
表 20-26 函数 RTC_Config_Mode_Enable.....	369
表 20-27 函数 RTC_Get_Operation_Off_Flag.....	369
表 20-28 函数 RTC_Get_Action_State_Flag.....	369
表 20-29 函数 RTC_Enable.....	370
表 20-30 函数 RTC_Alarm_A_Enable.....	370
表 20-31 函数 RTC_Alarm_A_Weekday_Enable.....	370

表 20-32 函数 RTC_Alarm_A_Weekday_Config.....	370
表 20-33 函数 RTC_Alarm_A_Hours_Enable.....	371
表 20-34 函数 RTC_Alarm_A_AMPM_Config.....	371
表 20-35 函数 RTC_Alarm_A_Hours_Config.....	371
表 20-36 函数 RTC_Alarm_A_Minutes_Enable.....	372
表 20-37 函数 RTC_Alarm_A_Minutes_Config.....	372
表 20-38 函数 RTC_Alarm_A_Seconds_Enable.....	372
表 20-39 函数 RTC_Alarm_A_Seconds_Config.....	372
表 20-40 函数 RTC_Alarm_B_Enable.....	373
表 20-41 函数 RTC_Alarm_B_Weekday_Enable.....	373
表 20-42 函数 RTC_Alarm_B_Weekday_Config.....	373
表 20-43 函数 RTC_Alarm_B_Hours_Enable.....	374
表 20-44 函数 RTC_Alarm_B_AMPM_Config.....	374
表 20-45 函数 RTC_Alarm_B_Hours_Config.....	374
表 20-46 函数 RTC_Alarm_B_Minutes_Enable.....	374
表 20-47 函数 RTC_Alarm_B_Minutes_Config.....	375
表 20-48 函数 RTC_Alarm_B_Seconds_Enable.....	375
表 20-49 函数 RTC_Alarm_B_Seconds_Config.....	375
表 20-50 函数 RTC_Weekday_Config.....	375
表 20-51 函数 RTC_AMPM_Config.....	376
表 20-52 函数 RTC_Hours_Config.....	376
表 20-53 函数 RTC_Minutes_Config.....	376
表 20-54 函数 RTC_Seconds_Config.....	377
表 20-55 函数 RTC_Year_Config.....	377
表 20-56 函数 RTC_Month_Config.....	377
表 20-57 函数 RTC_Day_Config.....	378
表 20-58 函数 RTC_Weekday_Backup_Config.....	378
表 20-59 函数 RTC_AMPM_Backup_Config.....	378
表 20-60 函数 RTC_Hours_Backup_Config.....	379
表 20-61 函数 RTC_Minutes_Backup_Config.....	379
表 20-62 函数 RTC_Seconds_Backup_Config.....	379
表 20-63 函数 RTC_Year_Backup_Config.....	380
表 20-64 函数 RTC_Month_Backup_Config.....	380
表 20-65 函数 RTC_Day_Backup_Config.....	380
表 20-66 函数 RTC_Timer1_Config.....	381
表 20-67 函数 RTC_Timer0_Config.....	381
表 20-68 函数 RTC_Timer1_Enable.....	381
表 20-69 函数 RTC_Timer0_Enable.....	381
表 20-70 函数 RTC_Timer1_Source_Config.....	382
表 20-71 函数 RTC_Timer0_Source_Config.....	382
表 20-72 函数 RTC_Time_Stamp_INT_Enable.....	383
表 20-73 函数 RTC_Time_Stamp_INT_Enable.....	383
表 20-74 函数 RTC_Timer1_INT_Enable.....	383
表 20-75 函数 RTC_Timer0_INT_Enable.....	383
表 20-76 函数 RTC_Time_Tick_INT_Enable.....	384

表 20-77 函数 RTC_Alarm_B_INT_Enable.....	384
表 20-78 函数 RTC_Alarm_A_INT_Enable.....	384
表 20-79 函数 RTC_Days_INT_Enable.....	384
表 20-80 函数 RTC_Hours_INT_Enable.....	385
表 20-81 函数 RTC_Minutes_INT_Enable.....	385
表 20-82 函数 RTC_Seconds_INT_Enable.....	385
表 20-83 函数 RTC_Get_Time_Stamp_INT_Flag.....	385
表 20-84 函数 RTC_Get_Time_Stamp_Overflow_INT_Flag.....	386
表 20-85 函数 RTC_Get_Timer1_INT_Flag.....	386
表 20-86 函数 RTC_Get_Timer0_INT_Flag.....	386
表 20-87 函数 RTC_Get_Timer0_INT_Flag.....	386
表 20-88 函数 RTC_Get_Alarm_B_INT_Flag.....	387
表 20-89 函数 RTC_Get_Alarm_A_INT_Flag.....	387
表 20-90 函数 RTC_Get_Days_INT_Flag.....	387
表 20-91 函数 RTC_Get_Hours_INT_Flag.....	388
表 20-92 函数 RTC_Get_Hours_INT_Flag.....	388
表 20-93 函数 RTC_Get_Seconds_INT_Flag.....	388
表 20-94 函数 RTC_Clear_Time_Stamp_INT_Flag.....	388
表 20-95 函数 RTC_Clear_Time_Stamp_Overflow_INT_Flag.....	389
表 20-96 函数 RTC_Clear_Timer1_INT_Flag.....	389
表 20-97 函数 RTC_Clear_Timer0_INT_Flag.....	389
表 20-98 函数 RTC_Clear_Time_Tick_INT_Flag.....	389
表 20-99 函数 RTC_Clear_Alarm_B_INT_Flag.....	390
表 20-100 函数 RTC_Clear_Alarm_A_INT_Flag.....	390
表 20-101 函数 RTC_Clear_Days_INT_Flag.....	390
表 20-102 函数 RTC_Clear_Hours_INT_Flag.....	390
表 20-103 函数 RTC_Clear_Minutes_INT_Flag.....	391
表 20-104 函数 RTC_Clear_Seconds_INT_Flag.....	391
表 21-1 SPI 寄存器结构说明.....	392
表 21-2 SPI 协议信息结构体说明.....	393
表 21-3 I2S 协议信息结构体说明.....	393
表 21-4 SPI 固件库函数列表.....	394
表 21-5 函数 SPI_Reset.....	395
表 21-6 函数 SPI_Configuration.....	395
表 21-7 函数 I2S_Configuration.....	396
表 21-8 函数 SPI_Struct_Init.....	396
表 21-9 函数 I2S_Struct_Init.....	396
表 21-10 函数 SPI_Cmd.....	396
表 21-11 函数 I2S_Mode_Select.....	397
表 21-12 函数 SPI_I2S_ReceiveData.....	397
表 21-13 函数 SPI_I2S_SendData32.....	397
表 21-14 函数 SPI_I2S_SendData8.....	397
表 21-15 函数 SPI_BaudRate_Config.....	398
表 21-16 函数 I2S_DIV_Config.....	398
表 21-17 函数 SPI_MODE_Config.....	398

表 21-18 函数 SPI_CLK_Config.....	399
表 21-19 函数 SPI_Data_Direction_Config.....	399
表 21-20 函数 SPI_Clock_Polarity_Config.....	399
表 21-21 函数 SPI_Clock_Edge_Config.....	400
表 21-22 函数 SPI_BIT_SELECT_Config.....	400
表 21-23 函数 SPI_I2S_MODE_Config.....	400
表 21-24 函数 SPI_I2S_STANDARD_Config.....	401
表 21-25 函数 SPI_PCM_Config.....	401
表 21-26 函数 SPI_CHLEN_Config.....	402
表 21-27 函数 SPI_PCM_CLOCK_Polarity_Config.....	402
表 21-28 函数 SPI_MAIN_CLOCK_OUT_Enable.....	402
表 21-29 函数 SPI_Receive_Overflow_INT_Enable.....	403
表 21-30 函数 SPI_Transmit_Overflow_INT_Enable.....	403
表 21-31 函数 SPI_RNEIE_INT_Enable.....	403
表 21-32 函数 SPI_TNEIE_INT_Enable.....	403
表 21-33 函数 SPI_Receive_DMA_INT_Enable.....	404
表 21-34 函数 SPI_Transmit_DMA_INT_Enable.....	404
表 21-35 函数 SPI_Transmit_CHSIDE_INT_Enable.....	404
表 21-36 函数 SPI_Get_BUSY_Flag.....	405
表 21-37 函数 SPI_Get_Receive_Overflow_Flag.....	405
表 21-38 函数 SPI_Get_Transmit_Overflow_Flag.....	405
表 21-39 函数 SPI_Get_Receive_Buf_Flag.....	405
表 21-40 函数 SPI_Get_Transmit_Buf_Flag.....	406
表 21-41 函数 SPI_Clear_Receive_Overflow_INT_Flag.....	406
表 21-42 函数 SPI_Clear_Transmit_Overflow_INT_Flag.....	406
表 22-1 SYSCTL 寄存器结构说明.....	407
表 22-2 SYSCTL 固件库函数列表.....	408
表 22-3 函数 SYSCTL_Get_V_Flag.....	409
表 22-4 函数 SYSCTL_Get_C_Flag.....	409
表 22-5 函数 SYSCTL_Get_Z_Flag.....	409
表 22-6 函数 SYSCTL_Get_N_Flag.....	409
表 22-7 函数 SYSCTL_Set_V_Flag.....	410
表 22-8 函数 SYSCTL_Set_C_Flag.....	410
表 22-9 函数 SYSCTL_Set_Z_Flag.....	410
表 22-10 函数 SYSCTL_Set_N_Flag.....	410
表 22-11 函数 SYSCTL_Sleep_On_Exit_Enable.....	411
表 22-12 函数 SYSCTL_Deep_Sleep_Enable.....	411
表 22-13 函数 SYSCTL_Interrupt_Awake_Enable.....	411
表 22-14 函数 SYSCTL_Stack_Align_State.....	411
表 22-15 函数 SYSCTL_Super_User_Config.....	412
表 22-16 函数 SYSCTL_Stack_Pointer_State.....	412
表 22-17 函数 SYSCTL_Stack_Pointer_Config.....	412
表 22-18 函数 SYSCTL_Exception_Reset_Enable.....	412
表 22-19 函数 SYSCTL_System_Reset_Enable.....	413
表 22-20 函数 SYSCTL_Vector_Offset_Config.....	413

表 22-21 函数 SYSCTL_Ram_Space_Config.....	413
表 22-22 函数 SYSCTL_Flash_Start_Remap_Config.....	414
表 23-1 SYSTICK 寄存器结构说明.....	415
表 23-2 SYSTICK 配置信息结构体说明.....	416
表 23-3 SYSTICK 固件库函数列表.....	416
表 23-4 函数 SYSTICK_Configuration.....	416
表 23-5 函数 SYSTICK_Cmd.....	417
表 23-6 函数 SYSTICK_Clock_Config.....	417
表 23-7 函数 SYSTICK_Systick_INT_Enable.....	417
表 23-8 函数 SYSTICK_Get_Count_Zero_Flag.....	418
表 23-9 函数 SYSTICK_Reload_Config.....	418
表 23-10 函数 SYSTICK_Counter_Update.....	418
表 23-11 函数 SYSTICK_Get_Reload.....	419
表 23-12 函数 SYSTICK_Get_Counter.....	419
表 24-1 USART 寄存器结构说明.....	421
表 24-2 USART 配置信息结构体说明.....	421
表 24-3 USART_7816 配置信息结构体说明.....	422
表 24-4 USART 固件库函数列表.....	423
表 24-5 函数 USART_Resett.....	426
表 24-6 函数 USART_Configuration.....	426
表 24-7 函数 USART_U7816R_Configuration.....	427
表 24-8 函数 USART_Struct_Init.....	427
表 24-9 函数 USART_U7816R_Struct_Init.....	427
表 24-10 函数 USART_Cmd.....	427
表 24-11 函数 USART_BaudRate_Clock_Config.....	428
表 24-12 函数 USART_HalfDuplex_ClockPolarity_Config.....	428
表 24-13 函数 USART_Transmit_Order_Config.....	429
表 24-14 函数 USART_Receive_Order_Config.....	429
表 24-15 函数 USART_Infrare_Detector_Voltage_Config.....	429
表 24-16 函数 USART_WeakUP_Enable.....	430
表 24-17 函数 USART_Clock_Source_Config.....	430
表 24-18 函数 USART_Address_Detection_Enable.....	430
表 24-19 函数 USART_Auto_BaudRate_Detection_Enable.....	431
表 24-20 函数 USART_Get_Auto_BaudRate_Detection_Flag.....	431
表 24-21 函数 USART_Send_Blank_Enable.....	431
表 24-22 函数 USART_SYN_Choice_Config.....	432
表 24-23 函数 USART_Transmit_Data_Enable.....	432
表 24-24 函数 USART_Receive_Data_Enable.....	432
表 24-25 函数 USART_STOP_Word_Config.....	433
表 24-26 函数 USART_Transmit_9Word_Select_Config.....	433
表 24-27 函数 USART_Parity_Select_Config.....	434
表 24-28 函数 USART_9Data_Enable.....	434
表 24-29 函数 USART_CTS_Enable.....	434
表 24-30 函数 USART_RTS_Enable.....	435
表 24-31 函数 USART_Infrare_Detector_Enable.....	435

表 24-32 函数 USART_RESHD_Enable.....	435
表 24-33 函数 USART_Singlet_Line_Mode_Enable.....	436
表 24-34 函数 USART_BaudRate_Integer_Config.....	436
表 24-35 函数 USART_BaudRate_Decimal1_Config.....	436
表 24-36 函数 USART_BaudRate_Decimal2_Config.....	437
表 24-37 函数 USART_SendData.....	437
表 24-38 函数 USART_TransmitData.....	437
表 24-39 函数 USART_ReceiveData.....	438
表 24-40 函数 USART_Address_Match_Config.....	438
表 24-41 函数 USART_Send_Idle_Frame_Enable.....	438
表 24-42 函数 USART_Receive_Idle_Frame_Config.....	439
表 24-43 函数 USART_7816_Cmd.....	439
表 24-44 函数 USART_7816_CLKOUT_Enable.....	439
表 24-45 函数 USART_7816_Error_Signal_Config.....	440
表 24-46 函数 USART_Passageway_Select_Config.....	440
表 24-47 函数 USART_BGT_Config.....	440
表 24-48 函数 USART_Transmit_Repeat_Enable.....	441
表 24-49 函数 USART_Receive_Repeat_Enable.....	441
表 24-50 函数 USART_Transmit_Repeat_Times_Config.....	442
表 24-51 函数 USART_Receive_Repeat_Times_Config.....	442
表 24-52 函数 USART_7816_CLKDIV_Config.....	442
表 24-53 函数 USART_7816_EGT_Config.....	443
表 24-54 函数 USART_7816_Resend_Mode_Select.....	443
表 24-55 函数 USART_Receive_Overflow_INT_Enable.....	444
表 24-56 函数 USART_Parity_ERROR_INT_Enable.....	444
表 24-57 函数 USART_Frame_ERROE_INT_Enable.....	444
表 24-58 函数 USART_Blank_INT_Enable.....	445
表 24-59 函数 USART_Auto_BaudRate_TimeOver_INT_Enable.....	445
表 24-60 函数 USART_WeakUP_INT_Enable.....	445
表 24-61 函数 USART_Transmit_ERROR_INT_Enable.....	446
表 24-62 函数 USART_Receive_ERROR_INT_Enable.....	446
表 24-63 函数 USART_CTS_INT_Enable.....	446
表 24-64 函数 USART_RDR_INT_Enable.....	447
表 24-65 函数 USART_TFE_INT_Enable.....	447
表 24-66 函数 USART_TXE_INT_Enable.....	447
表 24-67 函数 USART_Receive_DMA_INT_Enable.....	448
表 24-68 函数 USART_Transmit_DMA_INT_Enable.....	448
表 24-69 函数 USART_IDLE_INT_Enable.....	448
表 24-70 函数 USART_UADM_INT_Enable.....	449
表 24-71 函数 USART_Get_Receive_Overflow_Flag.....	449
表 24-72 函数 USART_Get_Parity_ERROR_Flag.....	449
表 24-73 函数 USART_Get_Frame_ERROR_Flag.....	450
表 24-74 函数 USART_Get_Blank_Flag.....	450
表 24-75 函数 USART_Get_Auto_Baudrate_TimeOver_Flag.....	450
表 24-76 函数 USART_Get_WeakUP_Flag.....	451

表 24-77 函数 USART_Get_7816Transmit_ERROR_Flag.....	451
表 24-78 函数 USART_Get_7816Receive_ERROR_Flag.....	451
表 24-79 函数 USART_Get_CTS_Flag.....	452
表 24-80 函数 USART_Get_Receive_BUFR_Ready_Flag.....	452
表 24-81 函数 USART_Get_WUEN_Flag.....	452
表 24-82 函数 USART_Get_Transmit_BUFR_Empty_Flag.....	453
表 24-83 函数 USART_Get_Transmitter_Empty_Flag.....	453
表 24-84 函数 USART_Get_Receive_Frame_Idel_Flag.....	453
表 24-85 函数 USART_Clear_Receive_Overflow_INT_Flag.....	454
表 24-86 函数 USART_Clear_Parity_ERROR_INT_Flag.....	454
表 24-87 函数 USART_Clear_Frame_ERROR_INT_Flag.....	454
表 24-88 函数 USART_Clear_Blank_INT_Flag.....	454
表 24-89 函数 USART_Clear_Auto_BaudRate_TimeOver_INT_Flag.....	455
表 24-90 函数 USART_Clear_WeakUP_INT_Flag.....	455
表 24-91 函数 USART_Clear_Transmit_ERROR_INT_Flag.....	455
表 24-92 函数 USART_Clear_Receive_ERROR_INT_Flag.....	456
表 24-93 函数 USART_Clear_CTS_INT_Flag.....	456
表 24-94 函数 USART_Clear_UADM_INT_Flag.....	456
表 24-95 函数 USART_Clear_IDLE_INT_Flag.....	456
表 24-96 函数 USART_Clear_Receive_BUFR_INT_Flag.....	457
表 24-97 函数 USART_Clear_Transmit_BUFR_INT_Flag.....	457
表 25-1 WWDT 寄存器结构说明.....	458
表 25-2 WWDT 固件库函数列表.....	459
表 25-3 函数 WWDT_Reset.....	459
表 25-4 函数 WWDT_Threshold_Config.....	459
表 25-5 函数 WWDT_Prescaler_Config.....	460
表 25-6 函数 WWDT_Enable.....	460
表 25-7 函数 WWDT_Counter_Config.....	461
表 25-8 函数 WWDT_Get_Counter.....	461
表 25-9 函数 WWDT_INT_Enable.....	461
表 25-10 函数 WWDT_Get_INT_Flag.....	461
表 25-11 函数 WWDT_Clear_INT_Flag.....	462

1 文档说明

1.1 简介

本文档为 KF32A 系列 MCU 固件库 API 使用手册。

该固件库包括了微控制器所有外设的驱动描述和应用实例。通过使用该固件库,用户无需深入掌握细节,可轻松应用每个外设,减少用户开发时间,降低开发成本。

该固件库代码符合“gnu99 标准”,固件函数通过校验传参实现实时错误检测,该动态检测适合于用户应用程序的开发和调试。但这会增加运行成本,可以在最终的应用代码中删去,达到优化代码大小和执行速度的目的。

该固件库是通用的,是用户学习如何设置 MCU 外设的一份参考资料,所以应用代码程序的大小和执行速度可能不是最优的。对于在代码大小和执行速度要求严格的场景,用户可根据实际需求对其进行调整。

本文档主要面向 KF32A 系列 MCU 应用开发者或爱好者,为其提供完整的固件库 API 使用说明。

更多详细信息请访问 ChipON 官网 www.chipon-ic.com。

1 外设缩写说明

表 1-1 外设缩写说明

序号	缩写	外设
1	ADC	模数转换器
2	ATIM	高级定时器
3	BKP	备份域
4	BTIM	基本定时器
5	CAN	控制器局域网总线
6	CCP	通用捕捉/比较/脉宽调制模块
7	CMP	模拟比较器模块
8	CRC	循环冗余单元
9	DMA	直接存储器
10	ECCP	增强型捕捉/比较/脉宽调制模块
11	FLASH	闪存
12	GPIO	通用输入/输出引脚
13	GPTIM	通用定时器
14	I2C	内部集成电路接口
15	INT	中断
16	IWDG	独立看门狗
17	OSC	振荡器
18	PCLK	外设模块时钟使能模块

19	PM	电源管理模块
20	RTC	实时时钟
21	SPI	串行外设接口
22	RST	复位模块
23	SYSCTL	系统控制
24	SYSTICK	节拍定时器
25	USART	通用同步/异步收发器
26	WWDG	窗口看门狗

1.2 命名规则

KF32A 系列 MCU 固件库遵循以下规则：

文件名均以“kf32a”作为开头，例如：kf32a_basic_adc.c。

函数名、外设寄存器结构体、外设寄存器地址、外设寄存器位域均以“外设”作为开头，例如：ADC_Reset(ADC_SFRmap* ADCx)。

1.3 编码规则

1.3.1 枚举型

在文件 KF32A_BASIC.h 中，枚举变量被定义如下：

表 1-2 外设缩写说明

枚举类型	枚举功能	枚举列表
FunctionalState;	功能使能控制状态	FALSE: 0 TRUE: 1
FlagStatus	状态标志	RESET: 0 SET: 1
INTStatus	中断状态	RESET: 0 SET: 1
RetStatus;	返回状态	FAILURE: 0 SUCCESS: 1
AbleStatus;	使能状态	DISABLE: 0 ENABLE: 1

1.3.2 结构体

在文件 KF32A_BASIC.h 中，包含了所有外设寄存器的结构体声明，下面以 GPIO 寄存器结构体声明为例：

```
typedef struct GPIO_MemMap
{
    volatile const uint32_t PIR;
    volatile      uint32_t POR;
    volatile      uint32_t PUR;
```

```
volatile      uint32_t PDR;
volatile      uint32_t PODR;
volatile      uint32_t PMOD;
volatile      uint32_t OMOD;
volatile      uint32_t LOCK;
volatile      uint32_t RMP[2];
volatile      uint32_t RESERVED[3];
volatile      uint32_t RMP_MSB;
}GPIO_SFRmap;
```

寄存器命名遵循上节的寄存器缩写命名规则，RESERVED[i]（i 为一个整数索引值）表示被保留区域。

1.3.3 宏定义

在文件 KF32A_BASIC.h 中，包含了所有外设寄存器入口地址、寄存器入口、位域的宏定义，下面以 GPIO 寄存器宏定义为例：

```
#define GPIOA_ADDR ((uint32_t)0x50000000)
GPIOA_ADDR 为 GPIOA 寄存器入口地址。
#define GPIOA_PIR (GPIOA_SFR->PIR)
GPIOA_PIR 为 GPIOA 的 PIR 寄存器入口。
#define GPIO_PIR_PXPIR0_POS (0)
GPIO_PIR_PXPIR0_POS 为 GPIOA 的 PIR 寄存器 PXPIR0 位域。
```

1.4 固件库描述

KF32A 系列 MCU 固件库包含如下所示文件目录：

```
|---Inc
    |---kf32a_basic_adc.h
    .
    .
|---src
    |---kf32a_basic_adc.c
    .
    .
|---system_init.c
|---system_init.h
```

表 1-3 固件库目录说明

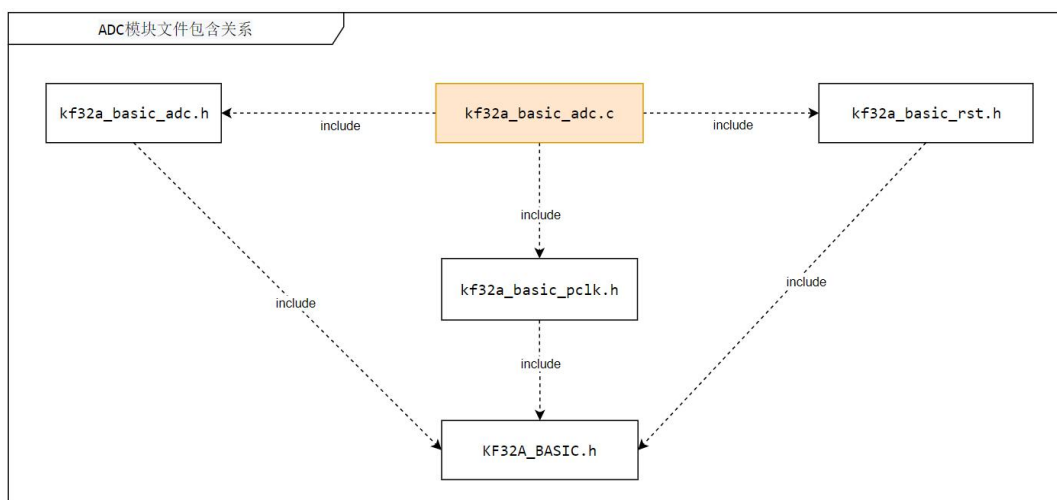
文件名	描述
inc	包含所有的外设寄存器固件库.h 头文件。
src	包含所有的外设寄存器固件库.c 代码文件。
system_init.c	提供了外设时钟与系统时钟初始化。
system_init.h	提供了外设库包含的头文件与系统时钟配置相关的宏定义。

2 模数转换模块（ADC）

模数转换模块，提供了一种 12 位逐次逼近型模拟数字转换器。它可测量 16 个常规通道和 4 个高优先级通道。各通道的 A/D 转换可以单次、连续模式执行，最高支持 20 个通道连续转换模式。ADC 的结果数据可以左对齐或右对齐方式存储在数据寄存器中。支持 DMA 触发、模拟看门狗、定时器触发及双 AD 模式。

2.1 文件引用关系

图 2-1 ADC 模块文件包含关系



2.2 ADC 寄存器结构

ADC 寄存器结构，ADC_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct ADC_MemMap
{
    volatile uint32_t    CTL0;
    volatile uint32_t    CTL1;
    volatile uint32_t    SCANSQ0;
    volatile uint32_t    SCANSQ1;
    volatile uint32_t    SCANSQ2;
    volatile uint32_t    HSCANSQ;
    volatile uint32_t    WDH;
    volatile uint32_t    WDL;
    volatile const uint32_t    DATA;
    volatile const uint32_t    HPDATA0;
    volatile const uint32_t    HPDATA1;
}
    
```

```
volatile const uint32_t    HPDATA2;
volatile const uint32_t    HPDATA3;
volatile uint32_t          HPOFF0;
volatile uint32_t          HPOFF1;
volatile uint32_t          HPOFF2;
volatile uint32_t          HPOFF3;
volatile uint32_t          SCANSQ3;
uint32_t                   RESERVED[2];
volatile uint32_t          STATE;
volatile uint32_t          DELAY;
} ADC_SFRmap;
```

表 2-1 ADC 寄存器结构说明

寄存器	描述
CTL0	ADC 控制寄存器 0,偏移:0x00
CTL1	ADC 控制寄存器 1,偏移:0x04
SCANSQ0	ADC 常规通道扫描序列设置寄存器 0,偏移:0x08
SCANSQ1	ADC 常规通道扫描序列设置寄存器 1,偏移:0x0C
SCANSQ2	ADC 常规通道扫描序列设置寄存器 2,偏移:0x10
HSCANSQ	ADC 高优先级通道扫描序列设置寄存器,偏移:0x14
WDH	ADC 模拟看门狗高阈值寄存器,偏移:0x18
WDL	ADC 模拟看门狗低阈值寄存器,偏移:0x1C
DATA	ADC 常规通道数据寄存器,偏移:0x20
HPDATA0	ADC 高优先级通道数据寄存器 0,偏移:0x24
HPDATA1	ADC 高优先级通道数据寄存器 1,偏移:0x28
HPDATA2	ADC 高优先级通道数据寄存器 2,偏移:0x2C
HPDATA3	ADC 高优先级通道数据寄存器 3,偏移:0x30
HPOFF0	ADC 高优先级通道数据偏移寄存器 0,偏移:0x34
HPOFF1	ADC 高优先级通道数据偏移寄存器 1,偏移:0x38
HPOFF2	ADC 高优先级通道数据偏移寄存器 2,偏移:0x3C
HPOFF3	ADC 高优先级通道数据偏移寄存器 3,偏移:0x40
SCANSQ3	ADC 常规通道扫描序列设置寄存器 3,偏移:0x44
RESERVED[2]	保留地址,偏移:0x48
STATE	ADC 状态寄存器,偏移:0x50
DELAY	ADC 状态寄存器,偏移:0x54

ADC0 快慢交叉延时寄存器结构, ADC_DELAY_SFRmap, 定义于文件 KF32A_BASIC.h 中, 如下:

```
typedef struct ADC_Delay_MemMap
{
    volatile uint32_t    DELAY;
} ADC_DELAY_SFRmap
```

表 2-2 ADC0 快慢交叉延时寄存器结构说明

寄存器	描述
DELAY	ADC0 快慢交叉延时寄存器,addr:0x400005D4

ADC 配置信息结构体，定义于文件 kf32a_basic_adc.h 中，如下：

```
typedef struct
{
    uint32_t    m_Clock;
    uint32_t    m_ClockDiv;
    FunctionalState    m_ScanMode;
    uint32_t    m_ContinuousMode;
    uint32_t    m_DataAlign;
    FunctionalState    m_ExternalTrig_EN;
    uint32_t    m_ExternalTrig;
    FunctionalState    m_HPEExternalTrig_EN;
    uint32_t    m_HPEExternalTrig;
    uint32_t    m_VoltageRef;
    uint32_t    m_NumOfConv;
    uint32_t    m_NumOfHPConv;
} ADC_InitTypeDef;
```

表 2-3 ADC 配置信息结构体说明

配置信息	描述
m_Clock	ADC 时钟源选择，取值为宏“ADC 时钟源”中的一个。
m_ClockDiv	ADC 时钟分频，取值为宏“ADC 时钟分频”中的一个。
m_ScanMode	ADC 扫描模式使能，取值为 TRUE 或 FALSE。
m_ContinuousMode	ADC 连续转换模式，取值为宏“ADC 连续转换”中的一个。
m_DataAlign	ADC 转换结果输出格式，取值为宏“ADC 转换结果对齐格式”中的一个。
m_ExternalTrig_EN	ADC 常规通道外部触发转换模式使能，取值为 TRUE 或 FALSE。
m_ExternalTrig	ADC 常规通道外部触发事件，取值为宏“ADC 常规通道外部触发事件”中的一个。
m_HPEExternalTrig_EN	ADC 高优先级通道外部触发转换模式使能，取值为 TRUE 或 FALSE。
m_HPEExternalTrig	高优先级通道外部触发事件，取值为宏“ADC 高优先级通道外部触发事件”中的一个。
m_VoltageRef	参考电压选择，取值为宏“ADC 参考电压选择”中的一个。
m_NumOfConv	ADC 常规通道扫描长度，取值为 1~16。
m_NumOfHPConv	ADC 高优先级通道扫描长度，取值为 1~4。

ADC 快慢交叉模式信息结构体，定义于文件 kf32a_basic_adc.h 中，如下：

```
typedef struct
```

```
{
    uint32_t    m_FastDelay;
    uint32_t    m_SlowDelay;
} ADC0_DELAY_InitTypeDef;
```

表 2-4 ADC 快慢交叉模式信息结构体说明

配置信息	描述
m_FastDelay	ADC 快速交叉模式延时时间，取值为 1~64。
m_SlowDelay	ADC 慢速交叉模式延时时间，取值为 1~64。

ADC 模拟看门狗信息结构体，定义于文件 kf32a_basic_adc.h 中，如下：

```
typedef struct
{
    uint32_t    m_WDSingleCH;
    FunctionalState    m_HPChannelWDEN;
    FunctionalState    m_ChannelWDEN;
    uint32_t    m_WDChannel;
    uint32_t    m_Threshold_H;
    uint32_t    m_Threshold_L;
} ADC_WD_InitTypeDef;
```

表 2-5 ADC 模拟看门狗信息结构体说明

配置信息	描述
m_WDSingleCH	ADC 模拟看门狗单通道使能，取值为宏“ADC 模拟看门狗单通道使能”中的一个。
m_HPChannelWDEN	ADC 高优先级通道上看门狗使能，取值为 TRUE 或 FALSE。
m_ChannelWDEN	ADC 常规通道上看门狗使能，取值为 TRUE 或 FALSE。
m_WDChannel	ADC 模拟看门狗通道选择，取值为宏“ADC 模拟看门狗通道选择”中的一个。
m_Threshold_H	ADC 模拟看门狗高阈值，取值为 0~0xFFFF。
m_Threshold_L	ADC 模拟看门狗低阈值，取值为 0~0xFFFF。

2.3 ADC 宏定义

ADC 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，ADC 其他相关宏定义详见文件 kf32a_basic_adc.h 及函数参数描述。

2.4 ADC 库函数

表 2-6 ADC 固件库函数列表

序号	函数名	描述
1	ADC_Reset	复位 ADC 外设，使能外设时钟。
2	ADC_Configuration	模数转换模块(ADC)初始化配置。

3	ADC_Struct_Init	初始化 ADC 配置信息结构体。
4	ADC_Delay_Configuration	ADC0_DELAY 初始化配置。
5	ADC_Delay_Struct_Init	初始化 ADC0 快慢交叉模式信息结构体。
6	ADC_Cmd	配置 ADC 工作使能。
7	ADC_Analog_Watchdog_Configuration	ADC 模拟看门狗初始化配置。
8	ADC_Analog_Watchdog_Struct_Init	初始化 ADC 模拟看门狗信息结构体。
9	ADC_Watchdog_Single_Channel_Enable	配置扫描模式中模拟看门狗单一通道使能。
10	ADC_Scan_Mode_Enable	配置扫描模式使能。
11	ADC_Bosssel_Comparator_Calibration	配置比较器的运放输入参考电压二分之一校准。
12	ADC_Bosssel_Calibration	配置参考电压二分之一运放电压校准。
13	ADC_Trim_Current_Intensity_Bias	配置偏置电流校准。
14	ADC_Analog_Clock_Config	配置 ADC 模拟部分时钟分频比选择。
15	ADC_Data_Align_Config	配置 A/D 转换结果输出格式。
16	ADC_Clock_Source_Config	配置 A/D 工作时钟源。
17	ADC-Regular_Channel_DMA_Cmd	配置直接存储器访问模式使能。
18	ADC_High_Priority_Channel_DMA_Cmd	配置直接存储器访问模式使能。
19	ADC_Double_Mode_Config	配置双 ADC 模式选择。
20	ADC_Reference_Voltage_Config	配置 ADC 参考电压选择。
21	ADC_Analog_Watchdog_Channel_Config	配置 ADC 模拟看门狗通道选择。
22	ADC-Regular_Channel_Watchdog_Enable	配置常规通道上看门狗使能。
23	ADC_External_Trig_Conv_Enable	配置常规通道外部触发转换模式使能。
24	ADC_External_Trig_Conv_Config	配置常规通道外部触发事件。
25	ADC-Regular_Channel_Config	配置常规通道扫描序列。
26	ADC-Regular-Sequencer_Length_Config	配置常规通道扫描长度。
27	ADC-Regular_Channel_TxCCRy_Trig_Enable	配置常规优先级通道的 Tx_CCRy 触发使能。
28	ADC_Software_Start_Conv	软件启动 A/D 常规通道转换。
29	ADC_Continuous_Mode_Cmd	配置连续转换使能。
30	ADC_Disc_Mode_Channel_Count_Config	配置间隔模式通道计数。
31	ADC_Disc_Mode_Cmd	配置常规通道上的间隔模式使能。
32	ADC_Get_Conversion_Value	获取常规通道转换结果数据。
33	ADC_High_Priority_Watchdog_Enable	配置高优先级通道上看门狗使能。

34	ADC_HPExternal_Trig_Conv_Enable	配置高优先级通道外部触发转换模式使能。
35	ADC_High_Priority_Channel_Config	配置高优先级通道扫描序列。
36	ADC_High_Priority_Sequencer_Length_Config	配置高优先级通道扫描长度。
37	ADC_Set_HPChannel_Conv_Value_Offset	配置高优先级通道转换结果数据偏移。
38	ADC_HPExternal_Trig_Conv_Config	配置高优先级通道外部触发事件。
39	ADC_Software_HPStart_Conv	软件启动 A/D 高优先级通道转换。
40	ADC_HPAuto_Conv_Cmd	配置自动高优先级通道组转换使能。
41	ADC_HPDisc_Mode_Cmd	配置高优先级通道上的间隔模式使能。
42	ADC_Get_HPConversion_Data	获取高优先级通道转换结果数据。
43	ADC_Set_INT_Enable	配置 ADC 中断使能。
44	ADC_Get_INT_Flag	获取 ADC 中断标志。
45	ADC_Clear_INT_Flag	清除 ADC 中断标志。
46	ADC_Get_INT_Status	获取 ADC 中断响应状态。

2.4.1 函数 ADC_Reset

表 2-7 函数 ADC_Reset

函数名	ADC_Reset
函数原型	void ADC_Reset (ADC_SFRmap* ADCx)
功能描述	复位 ADC 外设，使能外设时钟。
输入参数 1	ADCx:指向 ADC 内存结构的指针，取值为 ADC0_SFR~ADC2_SFR。
返回值	无
被调用函数 1	void RST_CTL1_Peripheral_Reset_Enable (uint32_t RST_CTL1Periph, FunctionalState NewState);
被调用函数 2	void PCLK_CTL1_Peripheral_Clock_Enable (uint32_t PCLK_CTL1_bit, FunctionalState NewState);

2.4.2 函数 ADC_Configuration

表 2-8 函数 ADC_Configuration

函数名	ADC_Configuration
函数原型	void ADC_Configuration (ADC_SFRmap* ADCx, ADC_InitTypeDef* adcInitStruct)
功能描述	模数转换模块(ADC)初始化配置。
输入参数 1	ADCx:指向 ADC 内存结构的指针，取值为 ADC0_SFR~ADC2_SFR。

输入参数 2	adcInitStruct: ADC 模块配置信息结构体指针。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.3 函数 ADC_Struct_Init

表 2-9 函数 ADC_Struct_Init

函数名	ADC_Struct_Init
函数原型	void ADC_Struct_Init (ADC_InitTypeDef* adcInitStruct)
功能描述	初始化 ADC 配置信息结构体。
输入参数 1	adcInitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

2.4.4 函数 ADC_Delay_Configuration

表 2-10 函数 ADC_Delay_Configuration

函数名	ADC_Delay_Configuration
函数原型	void ADC_Delay_Configuration (ADC0_DELAY_InitTypeDef* adc0Delay)
功能描述	ADC0_DELAY 初始化配置。
输入参数 1	adc0Delay: ADC0 快慢交叉模式信息结构体。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.5 函数 ADC_Delay_Struct_Init

表 2-11 函数 ADC_Delay_Struct_Init

函数名	ADC_Delay_Struct_Init
函数原型	void ADC_Delay_Struct_Init (ADC0_DELAY_InitTypeDef* adc0Delay)
功能描述	初始化 ADC0 快慢交叉模式信息结构体。
输入参数 1	adc0Delay: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

2.4.6 函数 ADC_Cmd

表 2-12 函数 ADC_Cmd

函数名	ADC_Cmd
函数原型	void ADC_Cmd (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置 ADC 工作使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: ADC 工作使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.7 函数 ADC_Analog_Watchdog_Configuration

表 2-13 函数 ADC_Analog_Watchdog_Configuration

函数名	ADC_Analog_Watchdog_Configuration
函数原型	void ADC_Analog_Watchdog_Configuration (ADC_SFRmap* ADCx, ADC_WD_InitTypeDef* adcAnalogWatchdog)
功能描述	ADC 模拟看门狗初始化配置。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	adcAnalogWatchdog: ADC 模拟看门狗信息结构体指针。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.8 函数 ADC_Analog_Watchdog_Struct_Init

表 2-14 函数 ADC_Analog_Watchdog_Struct_Init

函数名	ADC_Analog_Watchdog_Struct_Init
函数原型	void ADC_Analog_Watchdog_Struct_Init (ADC_WD_InitTypeDef* adcAnalogWatchdog)
功能描述	初始化 ADC 模拟看门狗信息结构体。
输入参数 1	adcAnalogWatchdog: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

2.4.9 函数 ADC_Watchdog_Single_Channel_Enable

表 2-15 函数 ADC_Watchdog_Single_Channel_Enable

函数名	ADC_Watchdog_Single_Channel_Enable
函数原型	void ADC_Watchdog_Single_Channel_Enable (ADC_SFRmap* ADCx,

	FunctionalState NewState)
功能描述	配置扫描模式中模拟看门狗单一通道使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 模拟看门狗单一通道使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.10 函数 ADC_Scan_Mode_Enable

表 2-16 函数 ADC_Scan_Mode_Enable

函数名	ADC_Scan_Mode_Enable
函数原型	void ADC_Scan_Mode_Enable (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置扫描模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 扫描模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.11 函数 ADC_Bosssel_Comparator_Calibration

表 2-17 函数 ADC_Bosssel_Comparator_Calibration

函数名	ADC_Bosssel_Comparator_Calibration
函数原型	void ADC_Bosssel_Comparator_Calibration (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置比较器的运放输入参考电压二分之一校准。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 校准配置状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.12 函数 ADC_Bosssel_Calibration

表 2-18 函数 ADC_Bosssel_Calibration

函数名	ADC_Bosssel_Calibration
函数原型	void ADC_Bosssel_Calibration (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置参考电压二分之一运放电压校准。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 校准配置状态, 取值范围为 TRUE 或 FALSE。

返回值	无
被调用函数	无

2.4.13 函数 ADC_Trim_Current_Intensity_Bias

表 2-19 函数 ADC_Trim_Current_Intensity_Bias

函数名	ADC_Trim_Current_Intensity_Bias
函数原型	void ADC_Trim_Current_Intensity_Bias (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置偏置电流校准。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 校准配置状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.14 函数 ADC_Analog_Clock_Config

表 2-20 函数 ADC_Analog_Clock_Config

函数名	ADC_Analog_Clock_Config
函数原型	void ADC_Analog_Clock_Config (ADC_SFRmap* ADCx, uint32_t ClockSelect)
功能描述	配置 ADC 模拟部分时钟分频比选择。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	ClockSelect: ADC 模拟部分时钟分频比, 取值范围为: ADC_CLK_DIV_1: T ADC_CLK_DIV_2: 1/2T ADC_CLK_DIV_4: 1/4T ADC_CLK_DIV_8: 1/8T ADC_CLK_DIV_16: 1/16T ADC_CLK_DIV_32: 1/32T ADC_CLK_500KHZ: 500KHz
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.15 函数 ADC_Data_Align_Config

表 2-21 函数 ADC_Data_Align_Config

函数名	ADC_Data_Align_Config
函数原型	void ADC_Data_Align_Config (ADC_SFRmap* ADCx, uint32_t DataAlign)

功能描述	配置 A/D 转换结果输出格式。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	DataAlign: ADC 转换结果对齐格式, 取值范围为: ADC_DATAALIGN_LEFT: 转换结果存储时左对齐 ADC_DATAALIGN_RIGHT: 转换结果存储时右对齐
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 16 函数 ADC_Clock_Source_Config

表 2-22 函数 ADC_Clock_Source_Config

函数名	ADC_Clock_Source_Config
函数原型	void ADC_Clock_Source_Config (ADC_SFRmap* ADCx, uint32_t ClockSource)
功能描述	配置 A/D 工作时钟源。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	ClockSource: ADC 工作时钟源, 取值范围为: ADC_SCLK: 选用 SCLK 作为 A/D 数字模块工作时钟 ADC_HFCLK: 选用 HFCLK 作为 A/D 数字模块工作时钟 ADC_LFCLK: 选用 LFCLK 作为 A/D 数字模块工作时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 17 函数 ADC_Regular_Channel_DMA_Cmd

表 2-23 函数 ADC_Regular_Channel_DMA_Cmd

函数名	ADC_Regular_Channel_DMA_Cmd
函数原型	void ADC_Regular_Channel_DMA_Cmd (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置直接存储器访问模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 直接存储器访问模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.18 函数 ADC_High_Priority_Channel_DMA_Cmd

表 2-24 函数 ADC_High_Priority_Channel_DMA_Cmd

函数名	ADC_High_Priority_Channel_DMA_Cmd
函数原型	void ADC_High_Priority_Channel_DMA_Cmd (ADC_SFRmap* ADCx, uint32_t HPChannel, FunctionalState NewState)
功能描述	配置直接存储器访问模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	HPChannel: 高优先级通道选择, 取值为: ADC_HPCH0: 高优先级通道 0 ADC_HPCH1: 高优先级通道 1 ADC_HPCH2: 高优先级通道 2 ADC_HPCH3: 高优先级通道 3
输入参数 3	NewState: 直接存储器访问模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.19 函数 ADC_Double_Mode_Config

表 2-25 函数 ADC_Double_Mode_Config

函数名	ADC_Double_Mode_Config
函数原型	void ADC_Double_Mode_Config (uint32_t WorkMode)
功能描述	配置双 ADC 模式选择。
输入参数 1	WorkMode: 双 ADC 模式, 取值范围为: ADC_SINGLE: 独立模式 ADC_REGULAR_HPRIORITY: 混合常规通道同步+高优先级通道同步模式 ADC_REGULAR_ALTERNATELY: 混合常规通道同步+交替触发模式 ADC_HPRIORITY_FAST_ALTERNATELY: 混合高优先级通道同步+快速交叉模式 ADC_HPRIORITY_SYNC: 高优先级同步模式 ADC_REGULAR_SYNC: 常规通道同步模式 ADC_FAST_ALTERNATELY_SYNC: 快速交叉同步 ADC_SLOW_ALTERNATELY_SYNC: 慢速交叉同步 ADC_ALTERNATELY_TRIGGER: 交替触发模式
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 20 函数 ADC_Reference_Voltage_Config

表 2-26 函数 ADC_Reference_Voltage_Config

函数名	ADC_Reference_Voltage_Config
函数原型	void ADC_Reference_Voltage_Config (ADC_SFRmap* ADCx, uint32_t RefVoltage)
功能描述	配置 ADC 参考电压选择。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	RefVoltage: 双 ADC 模式, 取值范围为: ADC_REF_INTERNAL: 内部参考电压作为 AD 转换电压 ADC_REF_VREF: Vref+作为 AD 转换电压 ADC_REF_AVDD: AVDD 作为 AD 转换电压
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 21 函数 ADC_Analog_Watchdog_Channel_Config

表 2-27 函数 ADC_Analog_Watchdog_Channel_Config

函数名	ADC_Analog_Watchdog_Channel_Config
函数原型	void ADC_Analog_Watchdog_Channel_Config (ADC_SFRmap* ADCx, uint32_t Channel)
功能描述	配置 ADC 模拟看门狗通道选择。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	Channel: ADC 模拟看门狗通道, 取值范围为 ADC_WDCH_0~ADC_WDCH_25。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 22 函数 ADC_Regular_Channel_Watchdog_Enable

表 2-28 函数 ADC_Regular_Channel_Watchdog_Enable

函数名	ADC_Regular_Channel_Watchdog_Enable
函数原型	void ADC_Regular_Channel_Watchdog_Enable (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置常规通道上看门狗使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 常规通道上看门狗使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无

被调用函数	无
-------	---

2.4.23 函数 ADC_External_Trig_Conv_Enable

表 2-29 函数 ADC_External_Trig_Conv_Enable

函数名	ADC_External_Trig_Conv_Enable
函数原型	void ADC_External_Trig_Conv_Enable (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置常规通道外部触发转换模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 常规通道外部触发转换模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.24 函数 ADC_External_Trig_Conv_Config

表 2-30 函数 ADC_External_Trig_Conv_Config

函数名	ADC_External_Trig_Conv_Config
函数原型	void ADC_External_Trig_Conv_Config (ADC_SFRmap* ADCx, uint32_t ExternalTrigEvent)
功能描述	配置常规通道外部触发事件。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	ExternalTrigEvent: 常规通道外部触发事件选择, 取值为: ADC_EXTERNALTRIG_CCPI1_CH1: CCP1 通道 1 ADC_EXTERNALTRIG_CCPI1_CH2: CCP1 通道 2 ADC_EXTERNALTRIG_CCPI1_CH3: CCP1 通道 3 ADC_EXTERNALTRIG_CCPI1_CH4: CCP1 通道 4 ADC_EXTERNALTRIG_CCPI2_CH1: CCP2 通道 1 ADC_EXTERNALTRIG_CCPI2_CH2: CCP2 通道 2 ADC_EXTERNALTRIG_CCPI2_CH3: CCP2 通道 3 ADC_EXTERNALTRIG_CCPI2_CH4: CCP2 通道 4 ADC_EXTERNALTRIG_CCPI3_CH1: CCP3 通道 1 ADC_EXTERNALTRIG_CCPI3_CH2: CCP3 通道 2 ADC_EXTERNALTRIG_CCPI3_CH3: CCP3 通道 3 ADC_EXTERNALTRIG_CCPI3_CH4: CCP3 通道 4 ADC_EXTERNALTRIG_CCPI4_CH1: CCP4 通道 1 ADC_EXTERNALTRIG_CCPI4_CH2: CCP4 通道 2 ADC_EXTERNALTRIG_CCPI4_CH3: CCP4 通道 3 ADC_EXTERNALTRIG_CCPI4_CH4: CCP4 通道 4 ADC_EXTERNALTRIG_CCPI18_CH1: CCP18 通道 1

ADC_EXTERNALTRIG_CCP18_CH2: CCP18 通道 2
ADC_EXTERNALTRIG_CCP18_CH3: CCP18 通道 3
ADC_EXTERNALTRIG_CCP18_CH4: CCP18 通道 4
ADC_EXTERNALTRIG_CCP19_CH1: CCP19 通道 1
ADC_EXTERNALTRIG_CCP19_CH2: CCP19 通道 2
ADC_EXTERNALTRIG_CCP19_CH3: CCP19 通道 3
ADC_EXTERNALTRIG_CCP19_CH4: CCP19 通道 4
ADC_EXTERNALTRIG_CCP20_CH1: CCP20 通道 1
ADC_EXTERNALTRIG_CCP20_CH2: CCP20 通道 2
ADC_EXTERNALTRIG_CCP20_CH3: CCP20 通道 3
ADC_EXTERNALTRIG_CCP20_CH4: CCP20 通道 4
ADC_EXTERNALTRIG_CCP21_CH1: CCP21 通道 1
ADC_EXTERNALTRIG_CCP21_CH2: CCP21 通道 2
ADC_EXTERNALTRIG_CCP21_CH3: CCP21 通道 3
ADC_EXTERNALTRIG_CCP21_CH4: CCP21 通道 4
ADC_EXTERNALTRIG_CCP5_CH1: CCP5 通道 1
ADC_EXTERNALTRIG_CCP5_CH2: CCP5 通道 2
ADC_EXTERNALTRIG_CCP5_CH3: CCP5 通道 3
ADC_EXTERNALTRIG_CCP5_CH4: CCP5 通道 4
ADC_EXTERNALTRIG_CCP22_CH1: CCP22 通道 1
ADC_EXTERNALTRIG_CCP22_CH2: CCP22 通道 2
ADC_EXTERNALTRIG_CCP22_CH3: CCP22 通道 3
ADC_EXTERNALTRIG_CCP22_CH4: CCP22 通道 4
ADC_EXTERNALTRIG_T1TRGO: T1TRGO
ADC_EXTERNALTRIG_T2TRGO: T2TRGO
ADC_EXTERNALTRIG_T3TRGO: T3TRGO
ADC_EXTERNALTRIG_T4TRGO: T4TRGO
ADC_EXTERNALTRIG_T18TRGO: T18TRGO
ADC_EXTERNALTRIG_T19TRGO: T19TRGO
ADC_EXTERNALTRIG_T20TRGO: T20TRGO
ADC_EXTERNALTRIG_T21TRGO: T21TRGO
ADC_EXTERNALTRIG_T5TRGO: T5TRGO
ADC_EXTERNALTRIG_T9TRGO: T9TRGO
ADC_EXTERNALTRIG_T14TRGO: T14TRGO
ADC_EXTERNALTRIG_T15TRGO: T15TRGO
ADC_EXTERNALTRIG_T5_OVERFLOW
ADC_EXTERNALTRIG_T6_OVERFLOW
ADC_EXTERNALTRIG_T9_OVERFLOW
ADC_EXTERNALTRIG_T10_OVERFLOW
ADC_EXTERNALTRIG_CCP9_CH1
ADC_EXTERNALTRIG_CCP9_CH2
ADC_EXTERNALTRIG_CCP9_CH3

	ADC_EXTERNALTRIG_CCP9_CH4 ADC_EXTERNALTRIG_EINT7: EINT7 ADC_EXTERNALTRIG_EINT15: EINT15 ADC_EXTERNALTRIG_CCP0_CH1 ADC_EXTERNALTRIG_CCP0_CH2 ADC_EXTERNALTRIG_CCP0_CH3 ADC_EXTERNALTRIG_CCP0_CH4 ADC_EXTERNALTRIG_CCP23_CH1 ADC_EXTERNALTRIG_CCP23_CH2 ADC_EXTERNALTRIG_CCP23_CH3 ADC_EXTERNALTRIG_CCP23_CH4
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 25 函数 ADC_Regular_Channel_Config

表 2-31 函数 ADC_Regular_Channel_Config

函数名	ADC_Regular_Channel_Config
函数原型	void ADC_Regular_Channel_Config (ADC_SFRmap* ADCx, uint32_t Channel, uint32_t Rank)
功能描述	配置常规通道扫描序列。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	Channel: ADC 通道选择, 取值范围为: ADC_CHANNEL_0: 通道 0--- ADC_CHANNEL_63: 通道 63
输入参数 3	Rank: 常规通道扫描 AD 输入的转换位置, 取值为 1~16。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 26 函数 ADC_Regular_Sequencer_Length_Config

表 2-32 函数 ADC_Regular_Sequencer_Length_Config

函数名	ADC_Regular_Sequencer_Length_Config
函数原型	void ADC_Regular_Sequencer_Length_Config (ADC_SFRmap* ADCx, uint32_t Length)
功能描述	配置常规通道扫描长度。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	Length: 常规通道扫描长度, 取值为 1~16。
返回值	无

被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);
-------	---

2.4.27 函数 ADC_Regular_Channel_TxCCRy_Trig_Enable

表 2-33 函数 ADC_Regular_Channel_TxCCRy_Trig_Enable

函数名	ADC_Regular_Channel_TxCCRy_Trig_Enable
函数原型	void ADC_Regular_Channel_TxCCRy_Trig_Enable (ADC_SFRmap* ADCx, uint32_t ExternalTrigEvent, FunctionalState NewState)
功能描述	配置常规优先级通道的 Tx_CCRy 触发使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	HPExternalTrigEvent: 常规优先级通道外部触发事件选择, 取值为: ADC_EXTERNALTRIG_T10_CCR0: 常规优先级通道的 T10_CCR0 触发使能 ADC_EXTERNALTRIG_T9_CCR1: 常规优先级通道的 T9_CCR1 触发使能 ADC_EXTERNALTRIG_T9_CCR0: 常规优先级通道的 T9_CCR0 触发使能 ADC_EXTERNALTRIG_T6_CCR0: 常规优先级通道的 T6_CCR0 触发使能 ADC_EXTERNALTRIG_T5_CCR1: 常规优先级通道的 T5_CCR1 触发使能 ADC_EXTERNALTRIG_T5_CCR0: 常规优先级通道的 T5_CCR0 触发使能
输入参数 3	NewState: ADC 中断使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.28 函数 ADC_Software_Start_Conv

表 2-34 函数 ADC_Software_Start_Conv

函数名	ADC_Software_Start_Conv
函数原型	void ADC_Software_Start_Conv(ADC_SFRmap* ADCx)
功能描述	软件启动 A/D 常规通道转换。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
返回值	无
被调用函数	无

2. 4. 29 函数 ADC_Continuous_Mode_Cmd

表 2-35 函数 ADC_Continuous_Mode_Cmd

函数名	ADC_Continuous_Mode_Cmd
函数原型	void ADC_Continuous_Mode_Cmd (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置连续转换使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 连续转换使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2. 4. 30 函数 ADC_Disc_Mode_Channel_Count_Config

表 2-36 函数 ADC_Disc_Mode_Channel_Count_Config

函数名	ADC_Disc_Mode_Channel_Count_Config
函数原型	void ADC_Disc_Mode_Channel_Count_Config (ADC_SFRmap* ADCx, uint8_t Number)
功能描述	配置间隔模式通道计数。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	Number: 间隔模式通道计数, 取值范围为 1~8。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2. 4. 31 函数 ADC_Disc_Mode_Cmd

表 2-37 函数 ADC_Disc_Mode_Cmd

函数名	ADC_Disc_Mode_Cmd
函数原型	void ADC_Disc_Mode_Cmd (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置常规通道上的间隔模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 常规通道上的间隔模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.32 函数 ADC_Get_Conversion_Value

表 2-38 函数 ADC_Get_Conversion_Value

函数名	ADC_Get_Conversion_Value
函数原型	uint16_t ADC_Get_Conversion_Value (ADC_SFRmap* ADCx)
功能描述	获取常规通道转换结果数据。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
返回值	常规通道转换结果数据, 16 位有效数据。
被调用函数	无

2.4.33 函数 ADC_High_Priority_Watchdog_Enable

表 2-39 函数 ADC_High_Priority_Watchdog_Enable

函数名	ADC_High_Priority_Watchdog_Enable
函数原型	void ADC_High_Priority_Watchdog_Enable (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置高优先级通道上看门狗使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 高优先级通道上看门狗使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.34 函数 ADC_HPEXternal_Trig_Conv_Enable

表 2-40 函数 ADC_HPEXternal_Trig_Conv_Enable

函数名	ADC_HPEXternal_Trig_Conv_Enable
函数原型	void ADC_HPEXternal_Trig_Conv_Enable (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置高优先级通道外部触发转换模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 高优先级通道外部触发转换模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.35 函数 ADC_High_Priority_Channel_Config

表 2-41 函数 ADC_High_Priority_Channel_Config

函数名	ADC_High_Priority_Channel_Config
-----	----------------------------------

函数原型	void ADC_High_Priority_Channel_Config (ADC_SFRmap* ADCx, uint32_t Channel, uint32_t Rank)
功能描述	配置高优先级通道扫描序列。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	Channel: ADC 通道选择, 取值范围为: ADC_CHANNEL_0: 通道 0--- ADC_CHANNEL_63: 通道 63
输入参数 3	Rank: 高优先级通道扫描 AD 输入的转换位置, 取值为 1~4。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.36 函数 ADC_High_Priority_Sequencer_Length_Config

表 2-42 函数 ADC_High_Priority_Sequencer_Length_Config

函数名	ADC_High_Priority_Sequencer_Length_Config
函数原型	void ADC_High_Priority_Sequencer_Length_Config (ADC_SFRmap* ADCx, uint32_t Length)
功能描述	配置高优先级通道扫描长度。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	Length: 高优先级通道扫描长度, 取值范围为 1~4。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.37 函数 ADC_Set_HPChannel_Conv_Value_Offset

表 2-43 函数 ADC_Set_HPChannel_Conv_Value_Offset

函数名	ADC_Set_HPChannel_Conv_Value_Offset
函数原型	void ADC_Set_HPChannel_Conv_Value_Offset (ADC_SFRmap* ADCx, uint32_t HPDoffChannel, uint32_t Offset)
功能描述	配置高优先级通道转换结果数据偏移。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	HPDoffChannel: 高优先级通道选择, 取值为: ADC_HPDOFF_0: 高优先级通道 0 ADC_HPDOFF_1: 高优先级通道 1 ADC_HPDOFF_2: 高优先级通道 2 ADC_HPDOFF_3: 高优先级通道 3
输入参数 3	Offset: 转换结果数据偏移, 取值为 12 位有效数据。
返回值	无
被调用函数	无

2. 4. 38 函数 ADC_HPEXternal_Trig_Conv_Config

表 2-44 函数 ADC_HPEXternal_Trig_Conv_Config

函数名	ADC_HPEXternal_Trig_Conv_Config
函数原型	void ADC_HPEXternal_Trig_Conv_Config (ADC_SFRmap* ADCx, uint32_t HPEXternalTrigEvent)
功能描述	配置高优先级通道外部触发事件。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	<p>HPEXternalTrigEvent: 高优先级通道外部触发事件选择, 取值为:</p> <p>ADC_HPEXTERNALTRIG_CCP1_CH1: CCP1 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP1_CH2: CCP1 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP1_CH3: CCP1 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP1_CH4: CCP1 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP2_CH1: CCP2 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP2_CH2: CCP2 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP2_CH3: CCP2 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP2_CH4: CCP2 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP3_CH1: CCP3 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP3_CH2: CCP3 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP3_CH3: CCP3 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP3_CH4: CCP3 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP4_CH1: CCP4 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP4_CH2: CCP4 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP4_CH3: CCP4 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP4_CH4: CCP4 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP18_CH1: CCP18 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP18_CH2: CCP18 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP18_CH3: CCP18 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP18_CH4: CCP18 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP19_CH1: CCP19 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP19_CH2: CCP19 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP19_CH3: CCP19 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP19_CH4: CCP19 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP20_CH1: CCP20 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP20_CH2: CCP20 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP20_CH3: CCP20 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP20_CH4: CCP20 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP21_CH1: CCP21 通道 1</p> <p>ADC_HPEXTERNALTRIG_CCP21_CH2: CCP21 通道 2</p> <p>ADC_HPEXTERNALTRIG_CCP21_CH3: CCP21 通道 3</p> <p>ADC_HPEXTERNALTRIG_CCP21_CH4: CCP21 通道 4</p> <p>ADC_HPEXTERNALTRIG_CCP5_CH1: CCP5 通道 1</p>

	ADC_HPEXTERNALTRIG_CCP5_CH2: CCP5 通道 2 ADC_HPEXTERNALTRIG_CCP5_CH3: CCP5 通道 3 ADC_HPEXTERNALTRIG_CCP5_CH4: CCP5 通道 4 ADC_HPEXTERNALTRIG_CCP22_CH1: CCP22 通道 1 ADC_HPEXTERNALTRIG_CCP22_CH2: CCP22 通道 2 ADC_HPEXTERNALTRIG_CCP22_CH3: CCP22 通道 3 ADC_HPEXTERNALTRIG_CCP22_CH4: CCP22 通道 4 ADC_HPEXTERNALTRIG_T1TRGO: T1TRGO ADC_HPEXTERNALTRIG_T2TRGO: T2TRGO ADC_HPEXTERNALTRIG_T3TRGO: T3TRGO ADC_HPEXTERNALTRIG_T4TRGO: T4TRGO ADC_HPEXTERNALTRIG_T18TRGO: T18TRGO ADC_HPEXTERNALTRIG_T19TRGO: T19TRGO ADC_HPEXTERNALTRIG_T20TRGO: T20TRGO ADC_HPEXTERNALTRIG_T21TRGO: T21TRGO ADC_HPEXTERNALTRIG_T5TRGO: T5TRGO ADC_HPEXTERNALTRIG_T9TRGO: T9TRGO ADC_HPEXTERNALTRIG_T14TRGO: T14TRGO ADC_HPEXTERNALTRIG_T15TRGO: T15TRGO ADC_HPEXTERNALTRIG_T5_OVERFLOW: T5 溢出 ADC_HPEXTERNALTRIG_T6_OVERFLOW: T6 溢出 ADC_HPEXTERNALTRIG_T9_OVERFLOW: T9 溢出 ADC_HPEXTERNALTRIG_T10_OVERFLOW: T10 溢出 ADC_HPEXTERNALTRIG_CCP9_CH1: CCP9 通道 1 ADC_HPEXTERNALTRIG_CCP9_CH2: CCP9 通道 2 ADC_HPEXTERNALTRIG_CCP9_CH3: CCP9 通道 3 ADC_HPEXTERNALTRIG_CCP9_CH4: CCP9 通道 4 ADC_HPEXTERNALTRIG_EINT7: EINT7 ADC_HPEXTERNALTRIG_EINT15: EINT15 ADC_HPEXTERNALTRIG_CCP0_CH1_CMP: CCP0_CH1 ADC_HPEXTERNALTRIG_CCP0_CH2_CMP: CCP0_CH2 ADC_HPEXTERNALTRIG_CCP0_CH3: CCP0_CH3 ADC_HPEXTERNALTRIG_CCP0_CH4: CCP0_CH4 ADC_HPEXTERNALTRIG_CCP23_CH1: CCP23 通道 1 ADC_HPEXTERNALTRIG_CCP23_CH2: CCP23 通道 2 ADC_HPEXTERNALTRIG_CCP23_CH3: CCP23 通道 3 ADC_HPEXTERNALTRIG_CCP23_CH4: CCP23 通道 4
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

2.4.39 函数 ADC_Software_HPStart_Conv

表 2-45 函数 ADC_Software_HPStart_Conv

函数名	ADC_Software_HPStart_Conv
函数原型	void ADC_Software_HPStart_Conv (ADC_SFRmap* ADCx)
功能描述	软件启动 A/D 高优先级通道转换。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
返回值	无
被调用函数	无

2.4.40 函数 ADC_HPAuto_Conv_Cmd

表 2-46 函数 ADC_HPAuto_Conv_Cmd

函数名	ADC_HPAuto_Conv_Cmd
函数原型	void ADC_HPAuto_Conv_Cmd (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置自动高优先级通道组转换使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 自动高优先级通道组转换使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.41 函数 ADC_HPDisc_Mode_Cmd

表 2-47 函数 ADC_HPDisc_Mode_Cmd

函数名	ADC_HPDisc_Mode_Cmd
函数原型	void ADC_HPDisc_Mode_Cmd (ADC_SFRmap* ADCx, FunctionalState NewState)
功能描述	配置高优先级通道上的间隔模式使能。
输入参数 1	ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	NewState: 高优先级通道上的间隔模式使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.42 函数 ADC_Get_HPConversion_Data

表 2-48 函数 ADC_Get_HPConversion_Data

函数名	ADC_Get_HPConversion_Data
-----	---------------------------

函数原型	uint16_t ADC_Get_HPConversion_Data (ADC_SFRmap* ADCx, uint8_t HPDataChannel)
功能描述	获取高优先级通道转换结果数据。
输入参数 1	输入 ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	HPDataChannel: 高优先级通道选择, 取值为: ADC_HPDATA_0: 高优先级通道 0 ADC_HPDATA_1: 高优先级通道 1 ADC_HPDATA_2: 高优先级通道 2 ADC_HPDATA_3: 高优先级通道 3
返回值	高优先级通道转换结果数据, 16 位有效数据。
被调用函数	无

2.4.43 函数 ADC_Set_INT_Enable

表 2-49 函数 ADC_Set_INT_Enable

函数名	ADC_Set_INT_Enable
函数原型	void ADC_Set_INT_Enable (ADC_SFRmap* ADCx, uint32_t InterruptType, FunctionalState NewState)
功能描述	配置 ADC 中断使能。
输入参数 1	输入 ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	InterruptType: ADC 中断类型, 取值为: ADC_INT_EOC: ADC 一次转换结束中断 ADC_INT_AWD: ADC 模拟看门狗中断 ADC_INT_HPEND: ADC 高优先级通道转换结束中断 ADC_INT_END: ADC 常规通道转换结束中断
输入参数 3	NewState: ADC 中断使能状态, 取值范围为 TRUE 或 FALSE。
返回值	无
被调用函数	无

2.4.44 函数 ADC_Get_INT_Flag

表 2-50 函数 ADC_Get_INT_Flag

函数名	ADC_Get_INT_Flag
函数原型	FlagStatus ADC_Get_INT_Flag (ADC_SFRmap* ADCx, uint32_t InterruptType)
功能描述	获取 ADC 中断标志。
输入参数 1	输入 ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	InterruptType: ADC 中断类型, 取值为: ADC_INT_EOC: ADC 一次转换结束中断 ADC_INT_AWD: ADC 模拟看门狗中断

	ADC_INT_HPENDING: ADC 高优先级通道转换结束中断 ADC_INT_END: ADC 常规通道转换结束中断
返回值	中断标志, 1: 发生了中断, 0: 未发生中断。
被调用函数	无

2.4.45 函数 ADC_Clear_INT_Flag

表 2-51 函数 ADC_Clear_INT_Flag

函数名	ADC_Clear_INT_Flag
函数原型	void ADC_Clear_INT_Flag (ADC_SFRmap* ADCx, uint32_t InterruptType)
功能描述	清除 ADC 中断标志。
输入参数 1	输入 ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	InterruptType: ADC 中断类型, 取值为: ADC_INT_EOC: ADC 一次转换结束中断 ADC_INT_AWD: ADC 模拟看门狗中断 ADC_INT_HPENDING: ADC 高优先级通道转换结束中断 ADC_INT_END: ADC 常规通道转换结束中断
返回值	无
被调用函数	无

2.4.46 函数 ADC_Get_INT_Status

表 2-52 函数 ADC_Get_INT_Status

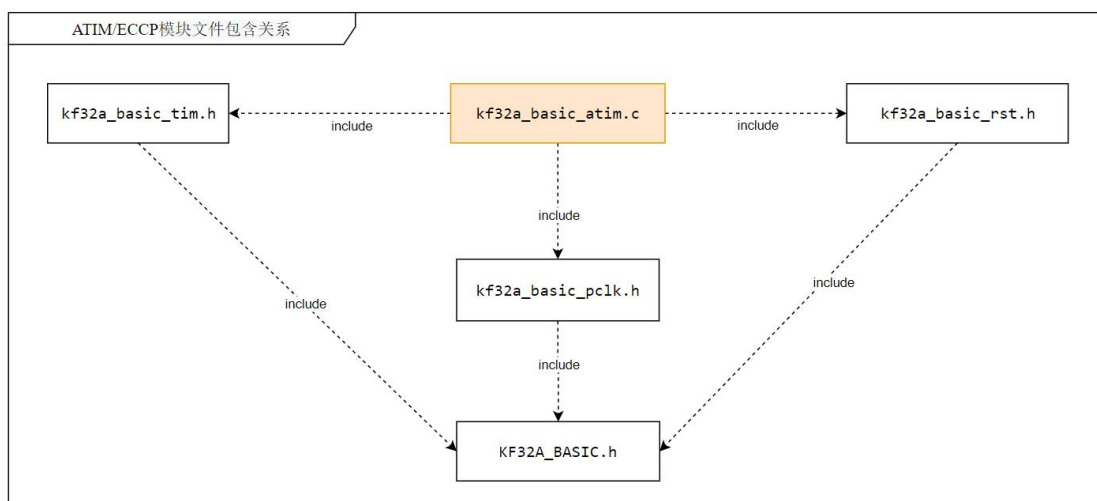
函数名	ADC_Get_INT_Status
函数原型	INTStatus ADC_Get_INT_Status (ADC_SFRmap* ADCx, uint32_t InterruptType)
功能描述	获取 ADC 中断响应状态。
输入参数 1	输入 ADCx: 指向 ADC 内存结构的指针, 取值为 ADC0_SFR~ADC2_SFR。
输入参数 2	InterruptType: ADC 中断类型, 取值为: ADC_INT_EOC: ADC 一次转换结束中断 ADC_INT_AWD: ADC 模拟看门狗中断 ADC_INT_HPENDING: ADC 高优先级通道转换结束中断 ADC_INT_END: ADC 常规通道转换结束中断
返回值	中断响应状态, 1: 发生待响应中断, 0: 未发生中断或未使能。
被调用函数	无

3 高级定时器(ATIM)

KungFu32 内核提供了两个 16 位的高级定时器（Advanced Timer，以下简称 ATIM）。高级定时器有 3 种计数方式：向上计数、向下计数和向上向下计数方式，可精确配置 1-65535 自由分频进行计数。

3.1 文件引用关系

图 3-1 ATIM/ECCP 模块文件包含关系



3.2 ATIM/ECCP 寄存器结构

ATIM/ECCP 寄存器结构，ATIM/ECCP_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct ATIM_MemMap
{
    volatile uint32_t    TXCNT;
    volatile uint32_t    TZCNT;
    volatile uint32_t    TXPPX;
    volatile uint32_t    TZPPZ;
    volatile uint32_t    TXPRSC;
    volatile uint32_t    TZPRSC;
    volatile uint32_t    TXCCR0;
    volatile uint32_t    TXCCR1;
    volatile uint32_t    TZCCR0;
    volatile uint32_t    TXCTL;
    volatile uint32_t    TZCTL;
}
    
```

```
volatile uint32_t    PXPCTL;
volatile uint32_t    PXASCTL;
uint32_t    RESERVED1[19];
volatile uint32_t    ECCPCTL1;
volatile uint32_t    ECCPXR1;
volatile uint32_t    ECCPXR2;
volatile uint32_t    ECCPXR3;
volatile uint32_t    ECCPXR4;
volatile uint32_t    PXUDCTL;
volatile uint32_t    ECCPCTL2;
volatile uint32_t    PXDTCTL;
volatile uint32_t    PWMXOC;
volatile uint32_t    PXATRCTL;
volatile uint32_t    PXASCTL0;
volatile uint32_t    PXASCTL1;
volatile uint32_t    ZPDCTL0;
volatile uint32_t    ZPDCTL1;
volatile uint32_t    ZPDPORT;
volatile uint32_t    ECCPXIE;
volatile uint32_t    ECCPXEGIF;
volatile const uint32_t    ECCPXDF;
volatile const uint32_t    ECCPXC1;
volatile const uint32_t    ECCPXC2;
volatile const uint32_t    ECCPXC3;
volatile const uint32_t    ECCPXC4;
uint32_t    RESERVED2;
volatile uint32_t    ECCPXSRI;
volatile uint32_t    ECCPXSRI;
volatile uint32_t    ECCPCTL3;
} ATIM_SFRmap, ECCP_SFRmap;
```

表 3-1 ATIM/ECCP 寄存器结构说明

寄存器	描述
TZCNT	Tx_CNT 寄存器, 偏移:0x00
TZCNT	Tz_CNT 寄存器, 偏移:0x04
TXPPX	Tx_PPx 周期寄存器, 偏移:0x08
TZPPZ	Tz_PPz 周期寄存器, 偏移:0x0C
TXPRSC	Tx 预分频设置寄存器, 偏移:0x10
TZPRSC	Tz 预分频设置寄存器, 偏移:0x14
TXCCR0	Tx 启动 ADC 设置寄存器 0, 偏移:0x18
TXCCR1	Tx 启动 ADC 设置寄存器 1, 偏移:0x1C
TZCCR0	Tz 启动 ADC 设置寄存器 0, 偏移:0x20
TXCTL	Tx 控制寄存器, 偏移:0x24

TZCTL	Tz 控制寄存器, 偏移:0x28
PXPDCTL	位置检测控制寄存器, 偏移:0x2C
PXASCTL	关断控制寄存器, 偏移:0x30
RESERVED1[19]	保留地址, 偏移:0x34
ECCPXCTL1	ECCP 控制寄存器 1, 偏移:0x80
ECCPXR1	ECCP 比较/PWM 寄存器 1, 偏移:0x84
ECCPXR2	ECCP 比较/PWM 寄存器 2, 偏移:0x88
ECCPXR3	ECCP 比较/PWM 寄存器 3, 偏移:0x8C
ECCPXR4	ECCP 比较/PWM 寄存器 4, 偏移:0x90
PXUDCTL	ECCP 更新控制寄存器, 偏移:0x94
ECCPXCTL2	ECCP 控制寄存器 2, 偏移:0x98
PXDTCTL	ECCP 死区控制寄存器, 偏移:0x9C
PWMXOC	ECCP PWMx 输出控制寄存器, 偏移:0xA0
PXATRCTL	ECCP PWMx 输出配置寄存器, 偏移:0xA4
PXASCTL0	ECCP PWMx 关断控制寄存器 0, 偏移:0xA8
PXASCTL1	ECCP PWMx 关断控制寄存器 1, 偏移:0xAC
ZPDCTL0	ECCP 零点检测控制寄存器 0, 偏移:0xB0
ZPDCTL1	ECCP 零点检测控制寄存器 1, 偏移:0xB4
ZPDPORT	ECCP 零点检测端口控制寄存器, 偏移:0xB8
ECCPXIE	ECCP 中断使能寄存器, 偏移:0xBC
ECCPXEGIF	ECCP 中断状态寄存器, 偏移:0xC0
TXUDTIM	Tx 更新计数器, 偏移:0xC4
TZUDTIM	Tz 更新计数器, 偏移:0xC8
ECCPXDF	触发 DMA 中断标志寄存器, 偏移:0xCC
ECCPXC1	捕捉寄存器 1, 偏移:0xD0
ECCPXC2	捕捉寄存器 2, 偏移:0xD4
ECCPXC3	捕捉寄存器 3, 偏移:0xD8
ECCPXC4	捕捉寄存器 4, 偏移:0xDC
RESERVED2	保留地址, 偏移:0xE0
ECCPXDE	触发 DMA 中断使能寄存器, 偏移:0xE4
ECCPXSIC	ECCP 中断清除寄存器, 偏移:0xE8
ECCPXCTL3	ECCP 控制寄存器 3, 偏移:0xEC

ATIM 配置信息结构体, ATIM_SFRmap, 定义于文件 kf32a_basic_tim.h 中, 如下:

```
typedef struct
{
    uint16_t    m_Counter;
    uint16_t    m_Period;
    uint16_t    m_Prescaler;
    uint16_t    m_Postscaler;
    uint16_t    m_CounterMode;
```

```
uint16_t    m_Clock;
uint16_t    m_WorkMode;
uint16_t    m_EXPulseSync;
}ATIM_InitTypeDef;
```

表 3-2 ATIM 配置信息结构体说明

寄存器	描述
m_Counter	定时器计数值，取值 16 位数据
m_Period	定时器周期值，取值 16 位数据
m_Prescaler	定时器预分频值，取值 16 位数据
m_Postscaler	定时器后分频器的值，取值为宏“ATIM 定时器后分频器的值”中的一个
m_CounterMode	定时器计数模式，取值为宏“ATIM 定时器计数模式”中的一个
m_Clock	定时器工作时钟，取值为宏“ATIM 定时器工作时钟”中的一个
m_WorkMode	定时/计数模式选择取值为宏“ATIM 定时/计数模式选择”中的一个
m_EXPulseSync	Tx 计数模式外部触发脉冲输入同步控制，取值为宏“ATIM 计数模式外部触发脉冲输入同步控制”中的一个

ECCP 捕捉模式配置信息结构体，ECCP_SFRmap，定义于文件 kf32a_basic_tim.h 中，如下：

```
typedef struct
{
    uint32_t    m_Channel;
    uint32_t    m_Mode;
    FunctionalState    m_PWMInput;
    FunctionalState    m_XORMode;
} ECCP_CaptureInitTypeDef;
```

表 3-3 ECCP 捕捉模式配置信息结构体说明

寄存器	描述
m_Channel	通道编号，取值范围满足“ECCP 通道”的宏
m_Mode	捕捉模式选择，取值范围满足宏 CHECK_ECCP_CAP_MODE 的约定条件
m_PWMInput	PWM 测量模式，取值为 TRUE 或 FALSE
m_XORMode	输入异或模式，取值为 TRUE 或 FALSE

ECCP PWM 模式配置信息结构体，ECCP_SFRmap，定义于文件 kf32a_basic_tim.h 中，如下：

```
typedef struct
{
    uint32_t    m_Channel;
    uint32_t    m_Mode;
```

```
uint16_t    m_DutyRatio;
uint8_t     m_DeadTime;
uint8_t     m_OutputMode;
uint16_t    m_HOutputCtl;
uint16_t    m_LOutputCtl;
FunctionalState m_PhaseMove;
FunctionalState m_SinglePWM;
FunctionalState m_CloseTimer;
}ECCP_PWMInitTypeDef;
```

表 3-4 ECCP 捕捉模式配置信息结构体说明

寄存器	描述
m_Channel	通道编号，取值范围满足“ECCP 通道”的宏
m_Mode	捕捉/比较器模式选择，取值范围满足宏 CHECK_ECCP_PWM_MODE 的约定条件
m_DutyRatio	占空比，取值范围为 0~65535
m_DeadTime	通道死区控制，取值为 0~255
m_OutputMode	通道输出模式，取值为“ECCP 通道输出模式”中的一个
m_HOutputCtl	通道 H 输出控制，取值为“ECCP 通道输出控制”中的一个
m_LOutputCtl	通道 L 输出控制，取值为“ECCP 通道输出控制”中的一个
m_PhaseMove	相位移动使能，取值为 TRUE 或 FALSE
m_SinglePWM	单脉冲输出模式，取值为 TRUE 或 FALSE
m_CloseTimer	单脉冲输出模式选择，取值为 TRUE 或 FALSE

3.3 ATIM/ECCP 宏定义

ATIM/ECCP 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，ATIM/ECCP 其他相关宏定义详见文件 kf32a_basic_tim.h 及函数参数描述。

3.4 ATIM/ECCP 库函数

表 3-5 ATIM 固件库函数列表

序号	函数名	描述
1	ATIM_X_Configuration	高级定时器(ATIM)配置，配置 Tx
2	ATIM_Z_Configuration	高级定时器(ATIM)配置，配置 Tz
3	ATIM_Struct_Init	初始化高级定时器配置信息结构体
4	ATIM_X_Cmd	定时器 Tx 启动控制使能
5	ATIM_Z_Cmd	定时器 Tz 启动控制使能
6	ATIM_X_Update_Configuration	高级定时器(ATIM)更新配置，配置 Tx
7	ATIM_Z_Update_Configuration	高级定时器(ATIM)更新配置，配置 Tz

8	ATIM_X_Updata_Cmd	定时器 Tx 更新使能控制
9	ATIM_Z_Updata_Cmd	定时器 Tz 更新使能控制
10	ATIM_X_Set_Counter	配置定时器 Tx 计数值
11	ATIM_Z_Set_Counter	配置定时器 Tz 计数值
12	ATIM_X_Set_Period	配置定时器 Tx 周期值
13	ATIM_Z_Set_Period	配置定时器 Tz 周期值
14	ATIM_X_Set_Prescaler	配置定时器 Tx 预分频值
15	ATIM_Z_Set_Prescaler	配置定时器 Tz 预分频值
16	ATIM_X_Counter_Mode_Config	配置定时器 Tx 计数模式
17	ATIM_Z_Counter_Mode_Config	配置定时器 Tz 计数模式
18	ATIM_X_Clock_Config	配置定时器 Tx 工作时钟
19	ATIM_Z_Clock_Config	配置定时器 Tz 工作时钟
20	ATIM_X_Postscaler_Config	配置定时器 Tx 的后分频器
21	ATIM_Z_Postscaler_Config	配置定时器 Tz 的后分频器
22	ATIM_X_External_Pulse_Sync_Config	配置定时器 Tx 计数模式外部触发脉冲输入同步控制位
23	ATIM_Z_External_Pulse_Sync_Config	配置定时器 Tz 计数模式外部触发脉冲输入同步控制位
24	ATIM_X_Work_Mode_Config	配置 Tx 定时/计数模式选择
25	ATIM_Z_Work_Mode_Config	配置 Tz 定时/计数模式选择
26	ATIM_X_Get_Direction	读 Tx 计数方向
27	ATIM_Z_Get_Direction	读 Tz 计数方向
28	ATIM_X_Overflow_AD_Enable	配置 Tx 上溢中断触发 AD 使能位
29	ATIM_Z_Overflow_AD_Enable	配置 Tz 上溢中断触发 AD 使能位
30	ATIM_X_Underflow_AD_Enable	配置 Tx 下溢中断触发 AD 使能位
31	ATIM_Z_Underflow_AD_Enable	配置 Tz 下溢中断触发 AD 使能位
32	ATIM_X_TriggerAD_Config	配置 Tx 溢出中断自动触发 AD 控制位
33	ATIM_Z_TriggerAD_Config	配置 Tz 溢出中断自动触发 AD 控制位
34	ATIM_X_Set_TriggerAD_Signal	配置定时器 Tx 触发 AD 寄存器的值
35	ATIM_Z_Set_TriggerAD_Signal	配置定时器 Tz 触发 AD 寄存器的值
36	ATIM_X_Updata_Immediately_Config	配置 Tx 立即更新控制位
37	ATIM_Z_Updata_Immediately_Config	配置 Tz 立即更新控制位
38	ATIM_X_Updata_Output_Ctl	配置 Tx 输出控制寄存器更新控制位
39	ATIM_Z_Updata_Output_Ctl	配置 Tz 输出控制寄存器更新控制位
40	ATIM_X_Updata_Enable	配置 Tx 更新使能
41	ATIM_Z_Updata_Enable	配置 Tz 更新使能
42	ATIM_X_Set_Updata_Counter	配置定时器 Tx 更新计数器
43	ATIM_Z_Set_Updata_Counter	配置定时器 Tz 更新计数器
44	ATIM_X_Slave_Mode_Config	配置 Tx 从模式选择位
45	ATIM_Z_Slave_Mode_Config	配置 Tz 从模式选择位
46	ATIM_Master_Mode_Config	配置 Tx 主模式选择位

47	ATIM_Master_Slave_Snyc_Enable	配置主从模式同步位
48	ATIM_Trigger_Select_Config	配置触发选择位
49	ATIM_Timer_Unite_Enable	配置 Tx、Tz 联立使能
50	ATIM_X_Get_Counter	读 Tx 定时器计数值
51	ATIM_Z_Get_Counter	读 Tz 定时器计数值
52	ATIM_X_Get_Period	读 Tx 定时器周期值
53	ATIM_Z_Get_Period	读 Tz 定时器周期值
54	ATIM_X_Get_Prescaler	读 Tx 定时器预分频值
55	ATIM_Z_Get_Prescaler	读 Tz 定时器预分频值
56	ATIM_X_Update_INT_Enable	配置 Tx 更新事件中断使能
57	ATIM_Z_Update_INT_Enable	配置 Tz 更新事件中断使能
58	ATIM_X_Overflow_INT_Enable	配置 Tx 计数溢出中断使能
59	ATIM_Z_Overflow_INT_Enable	配置 Tz 计数溢出中断使能
60	ATIM_X_Trigger_INT_Enable	配置 Tx 触发事件中断使能
61	ATIM_X_Update_DMA_Enable	配置 Tx 更新事件的 DMA 请求使能
62	ATIM_Z_Update_DMA_Enable	配置 Tz 更新事件的 DMA 请求使能
63	ATIM_X_Trigger_DMA_Enable	允许 Tx 触发事件的 DMA 请求配置
64	ATIM_X_Get_Update_INT_Flag	读取 Tx 更新事件中断标志
65	ATIM_Z_Get_Update_INT_Flag	读取 Tz 更新事件中断标志
66	ATIM_X_Get_Overflow_INT_Flag	读取 Tx 计数溢出中断标志
67	ATIM_Z_Get_Overflow_INT_Flag	读取 Tz 计数溢出中断标志
68	ATIM_X_Get_Trigger_INT_Flag	读取 Tx 触发事件中断标志
69	ATIM_X_Generate_Trigger_Config	产生触发事件配置位
70	ATIM_X_Get_Update_DMA_INT_Flag	读取 Tx 更新事件 DMA 中断标志
71	ATIM_Z_Get_Update_DMA_INT_Flag	读取 Tz 更新事件 DMA 中断标志
72	ATIM_X_Get_Trigger_DMA_INT_Flag	读取 Tx 触发事件触发 DMA 中断标志
73	ATIM_X_Clear_Update_INT_Flag	清除 Tx 更新事件中断标志
74	ATIM_Z_Clear_Update_INT_Flag	清除 Tz 更新事件中断标志
75	ATIM_X_Clear_Overflow_INT_Flag	清除 Tx 溢出中断标志
76	ATIM_Z_Clear_Overflow_INT_Flag	清除 Tz 溢出中断标志
77	ATIM_X_Clear_Trigger_INT_Flag	清除 Tx 触发事件中断标志

表 3-6 ECCP 固件库函数列表

序号	函数名	描述
78	ECCP_Compare_Configuration	增强型比较器(ECCP)配置
79	ECCP_Capture_Configuration	增强型捕捉模块(ECCP)配置
80	ECCP_Capture_Struct_Init	初始化增强型捕捉模块(ECCP)配置信息结构体
81	ECCP_PWM_Configuration	增强型 PWM 模块(ECCP)配置
82	ECCP_PWM_Struct_Init	初始化增强型 PWM 模块(ECCP)配置信息结构体

		构体
83	ECCP_Capture_Mode_Config	配置 ECCP 捕捉功能
84	ECCP_Compare_Mode_Config	配置 ECCP 比较功能
85	ECCP_PWM_Mode_Config	配置 ECCP PWM 功能,模式选择, 自由, 协同, 单时机模式
86	ECCP_Get_Capture_Result	读取 ECCP 捕捉寄存器
87	ECCP_Get_Compare_Result	读取 ECCP 比较/EPWM 占空比寄存器
88	ECCP_Set_Compare_Result	配置 ECCP 比较/PWM 占空比寄存器
89	ECCP_Generate_Trigger_Config	产生捕捉/比较事件配置位
90	ECCP_PWM_Input_Enable	PWM 输入测量模式使能
91	ECCP_Input_XOR_Enable	输入异或使能位配置
92	ECCP_Single_Pulse_Enable	配置单脉冲输出模式
93	ECCP_Single_Pulse_Shut_Enable	配置单脉冲输出模式选择
94	ECCP_PWM_Restart_Enable	配置 PWM 通道 1/2/3/4 重启使能位
95	ECCP_Dead_Time_Config	配置 PWM 通道死区延时时间
96	ECCP_Channel_Output_Control	配置 ECCP 通道输出控制
97	ECCP_Channel_Output_Mode	配置 ECCP 通道输出模式
98	ECCP_Channel_Work_State_Config	配置通道自动关闭事件状态位
99	ECCP_Get_Channel_Work_State	读取通道自动关闭事件状态位
100	ECCP_CHANNEL4_Shutdown_SEL	配置通道自动关闭源选择位
101	ECCP_Channel_Shutdown_Signal	配置通道自动关闭源选择位
102	ECCP_Channel_Pin_Ctl	配置引脚 ECCPxCHyH/ECCPxCHyL 关闭状态控制位
103	ECCP_Zero_Clock_Config	配置零点检测时钟
104	ECCP_Channel_Pin_Tristate_Enable	配置引脚 ECCPxCHyH/ECCPxCHyL 三态控制位
105	ECCP_Channel_INT_Enable	通道捕获/比较的中断使能配置
106	ECCP_X_Turn_off_DMA_Enable	关断事件的 DMA 请求使能位
107	ECCP_Channel_DMA_Enable	通道捕获/比较的 DMA 请求允许使能位配置
108	ECCP_Get_Channel_Trigger_INT_Flag	读取 ECCP 通道捕捉/比较中断标志
109	ECCP_X_Get_Turn_off_DMA_Flag	读取关断事件触发 DMA 标志
110	ECCP_Get_Trigger_DMA_INT_Flag	读取 ECCP 通道捕捉/比较 DMA 中断标志
111	ECCP_Clear_Channel_INT_Flag	清除 ECCPx 通道捕捉/比较中断标志
112	ECCP_PWM_Move_Phase_Enable	PWM 相位移动使能位配置
113	ECCP_Channel_Zero_Detect_Sequential_Ctl	配置通道的零点检测时序控制
114	ECCP_Get_Channel_Zero_Detection_State	读取通道零点检测的感应电压检测状态
115	ECCP_Clear_Channel_Zero_Detection_State	清零通道零点检测的感应电压检测状态
116	ECCP_Channel_Zero_Detect_Enable	配置通道的零点检测比较器使能位

117	ECCP_Channel_Zero_Voltage_Config	配置通道零点检测电压选择位
-----	----------------------------------	---------------

3.4.1 函数 ATIM_X_Configuration

表 3-7 函数 ATIM_X_Configuration

函数名	ATIM_X_Configuration
函数原型	void ATIM_X_Configuration(ATIM_SFRmap* ATIMx, ATIM_InitTypeDef* atimInitStruct)
功能描述	高级定时器(ATIM)配置，配置 Tx
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T5_SFR/T9_SFR
输入参数 2	atimInitStruct: 高级定时器配置信息结构体指针
返回值	无
被调用函数	无

3.4.2 函数 ATIM_Z_Configuration

表 3-8 函数 ATIM_Z_Configuration

函数名	ATIM_Z_Configuration
函数原型	void ATIM_Z_Configuration(ATIM_SFRmap* ATIMx, ATIM_InitTypeDef* atimInitStruct)
功能描述	高级定时器(ATIM)配置，配置 Tz
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T6_SFR/T10_SFR
输入参数 2	atimInitStruct: 高级定时器配置信息结构体指针
返回值	无
被调用函数	无

3.4.3 函数 ATIM_Struct_Init

表 3-9 函数 ATIM_Struct_Init

函数名	ATIM_Struct_Init
函数原型	void ATIM_Struct_Init(ATIM_InitTypeDef* atimInitStruct)
功能描述	初始化高级定时器配置信息结构体
输入参数 1	atimInitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

3.4.4 函数 ATIM_X_Cmd

表 3-10 函数 ATIM_X_Cmd

函数名	ATIM_X_Cmd
函数原型	void ATIM_X_Cmd (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	定时器 Tx 启动控制使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: 定时器使能控制, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.5 函数 ATIM_Z_Cmd

表 3-11 函数 ATIM_Z_Cmd

函数名	ATIM_Z_Cmd
函数原型	void ATIM_Z_Cmd (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	定时器 Tz 启动控制使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	atimInitStruct: 高级定时器配置信息结构体指针
返回值	无
被调用函数	无

3.4.6 函数 ATIM_X_Updata_Configuration

表 3-12 函数 ATIM_X_Updata_Configuration

函数名	ATIM_X_Updata_Configuration
函数原型	void ATIM_X_Updata_Configuration (ATIM_SFRmap* ATIMx,uint8_t UpdataCounter,uint32_t UpdataOutput,uint32_t UpdataImmediately)
功能描述	高级定时器(ATIM)更新配置, 配置 Tx
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	UpCounter: 高级定时器更新计数器的值, 取值 8 位数据 data
输入参数 3	UpdataOutput: 更新输出控制寄存器使能控制, 取值为 TRUE 或 FALSE
输入参数 4	UpdataImmediately: 立即更新定时器使能控制, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.7 函数 ATIM_Z_Updata_Configuration

表 3-13 函数 ATIM_Z_Updata_Configuration

函数名	ATIM_Z_Updata_Configuration
函数原型	void ATIM_Z_Updata_Configuration (ATIM_SFRmap* ATIMx,uint8_t

	UpdateCounter,uint32_t UpdateOutput,uint32_t UpdateImmediately)
功能描述	高级定时器(ATIM)更新配置, 配置 Tz
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	UpCounter: 高级定时器更新计数器的值, 取值 8 位数据 data
输入参数 3	UpdateOutput: 更新输出控制寄存器使能控制, 取值为 TRUE 或 FALSE
输入参数 4	UpdateImmediately: 立即更新定时器使能控制, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.8 函数 ATIM_X_Update_Cmd

表 3-14 函数 ATIM_X_Update_Cmd

函数名	ATIM_X_Update_Cmd
函数原型	void ATIM_X_Update_Cmd (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	定时器 Tx 更新使能控制
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: 定时器 Tx 更新使能控制, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.9 函数 ATIM_Z_Update_Cmd

表 3-15 函数 ATIM_Z_Update_Cmd

函数名	ATIM_Z_Update_Cmd
函数原型	void ATIM_Z_Update_Cmd (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	定时器 Tz 更新使能控制
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewState: 定时器 Tz 更新使能控制, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.10 函数 ATIM_X_Set_Counter

表 3-16 函数 ATIM_X_Set_Counter

函数名	ATIM_X_Set_Counter
函数原型	void ATIM_X_Set_Counter (ATIM_SFRmap* ATIMx, uint16_t Counter)
功能描述	配置定时器 Tx 计数值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR

输入参数 2	Counter: 计数值, 取值 16 位数据
返回值	无
被调用函数	无

3.4.11 函数 ATIM_Z_Set_Counter

表 3-17 函数 ATIM_Z_Set_Counter

函数名	ATIM_Z_Set_Counter
函数原型	void ATIM_Z_Set_Counter (ATIM_SFRmap* ATIMx, uint16_t Counter)
功能描述	配置定时器 Tz 计数值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T6_SFR
输入参数 2	Counter: 计数值, 取值 16 位数据
返回值	无
被调用函数	无

3.4.12 函数 ATIM_X_Set_Period

表 3-18 函数 ATIM_X_Set_Period

函数名	ATIM_X_Set_Period
函数原型	void ATIM_X_Set_Period (ATIM_SFRmap* ATIMx, uint16_t Period)
功能描述	配置定时器 Tx 周期值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	Period: 周期值, 取值 16 位数据
返回值	无
被调用函数	无

3.4.13 函数 ATIM_Z_Set_Period

表 3-19 函数 ATIM_Z_Set_Period

函数名	ATIM_Z_Set_Period
函数原型	void ATIM_Z_Set_Period (ATIM_SFRmap* ATIMx, uint16_t Period)
功能描述	配置定时器 Tz 周期值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	Period: 周期值, 取值 16 位数据
返回值	无
被调用函数	无

3. 4. 14 函数 ATIM_X_Set_Prescaler

表 3-20 函数 ATIM_X_Set_Prescaler

函数名	ATIM_X_Set_Prescaler
函数原型	void ATIM_X_Set_Prescaler (ATIM_SFRmap* ATIMx, uint16_t Prescaler)
功能描述	配置定时器 Tx 预分频值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	Prescaler: 预分频, 取值 16 位数据
返回值	无
被调用函数	无

3. 4. 15 函数 ATIM_Z_Set_Prescaler

表 3-21 函数 ATIM_Z_Set_Prescaler

函数名	ATIM_Z_Set_Prescaler
函数原型	void ATIM_Z_Set_Prescaler (ATIM_SFRmap* ATIMx, uint16_t Prescaler)
功能描述	配置定时器 Tz 预分频值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	Prescaler: 预分频, 取值 16 位数据
返回值	无
被调用函数	无

3. 4. 16 函数 ATIM_X_Counter_Mode_Config

表 3-22 函数 ATIM_X_Counter_Mode_Config

函数名	ATIM_X_Counter_Mode_Config
函数原型	void ATIM_X_Counter_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t CounterMode)
功能描述	配置定时器 Tx 计数模式
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	CounterMode: 新的计数模式, 取值范围为: ATIM_COUNT_DOWN_UF: 向下计数,下溢产生中断标志 ATIM_COUNT_UP_OF: 向上计数,上溢产生中断标志 ATIM_COUNT_UP_DOWN_OF: 向上-向下计数,上溢产生中断标志 ATIM_COUNT_UP_DOWN_UF: 向上-向下计数,下溢产生中断标志 ATIM_COUNT_UP_DOWN_OUF: 向上-向下计数,上溢和下溢产生中断标志
返回值	无
被调用函数	无

3. 4. 17 函数 ATIM_Z_Counter_Mode_Config

表 3-23 函数 ATIM_Z_Counter_Mode_Config

函数名	ATIM_Z_Counter_Mode_Config
函数原型	void ATIM_Z_Counter_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t CounterMode)
功能描述	配置定时器 Tz 计数模式
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	CounterMode: 新的计数模式, 取值范围为: ATIM_COUNT_DOWN_UF: 向下计数,下溢产生中断标志 ATIM_COUNT_UP_OF: 向上计数,上溢产生中断标志 ATIM_COUNT_UP_DOWN_OF: 向上-向下计数,上溢产生中断标志 ATIM_COUNT_UP_DOWN_UF: 向上-向下计数,下溢产生中断标志 ATIM_COUNT_UP_DOWN_OUF: 向上-向下计数,上溢和下溢产生中断标志
返回值	无
被调用函数	无

3. 4. 18 函数 ATIM_X_Clock_Config

表 3-24 函数 ATIM_X_Clock_Config

函数名	ATIM_X_Clock_Config
函数原型	void ATIM_X_Clock_Config (ATIM_SFRmap* ATIMx, uint32_t NewClock)
功能描述	配置定时器 Tx 工作时钟
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewClock: 定时器 Tx 工作时钟, 取值范围为: ATIM_SCLK: 选用 SCLK 时钟 ATIM_HFCLK: 选用 HFCLK 时钟 ATIM_LFCLK: 选用 LFCLK 时钟
返回值	无
被调用函数	无

3. 4. 19 函数 ATIM_Z_Clock_Config

表 3-25 函数 ATIM_Z_Clock_Config

函数名	ATIM_Z_Clock_Config
函数原型	void ATIM_Z_Clock_Config (ATIM_SFRmap* ATIMx, uint32_t NewClock)
功能描述	配置定时器 Tz 工作时钟
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewClock: 定时器 Tz 工作时钟, 取值范围为: ATIM_SCLK: 选用 SCLK 时钟

	ATIM_HFCLK: 选用 HFCLK 时钟 ATIM_LFCLK: 选用 LFCLK 时钟
返回值	无
被调用函数	无

3.4.20 函数 ATIM_X_Postscaler_Config

表 3-26 函数 ATIM_X_Postscaler_Config

函数名	ATIM_X_Postscaler_Config
函数原型	void ATIM_X_Postscaler_Config (ATIM_SFRmap* ATIMx, uint32_t NewPostscaler)
功能描述	配置定时器 Tx 的后分频器
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewPostscaler: 定时器 Tx 的后分频器, 取值为 4 位数据, 范围如下: ATIM_POSTSCALER_DIV_1: 分频比为 1:1 ATIM_POSTSCALER_DIV_2: 分频比为 1:2 ATIM_POSTSCALER_DIV_3: 分频比为 1:3 ATIM_POSTSCALER_DIV_4: 分频比为 1:4 ATIM_POSTSCALER_DIV_5: 分频比为 1:5 ATIM_POSTSCALER_DIV_6: 分频比为 1:6 ATIM_POSTSCALER_DIV_7: 分频比为 1:7 ATIM_POSTSCALER_DIV_8: 分频比为 1:8 ATIM_POSTSCALER_DIV_9: 分频比为 1:9 ATIM_POSTSCALER_DIV_10: 分频比为 1:10 ATIM_POSTSCALER_DIV_11: 分频比为 1:11 ATIM_POSTSCALER_DIV_12: 分频比为 1:12 ATIM_POSTSCALER_DIV_13: 分频比为 1:13 ATIM_POSTSCALER_DIV_14: 分频比为 1:14 ATIM_POSTSCALER_DIV_15: 分频比为 1:15 ATIM_POSTSCALER_DIV_16: 分频比为 1:16
返回值	无
被调用函数	无

3.4.21 函数 ATIM_Z_Postscaler_Config

表 3-27 函数 ATIM_Z_Postscaler_Config

函数名	ATIM_Z_Postscaler_Config
函数原型	void ATIM_Z_Postscaler_Config (ATIM_SFRmap* ATIMx, uint32_t NewPostscaler)
功能描述	配置定时器 Tz 的后分频器
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR

输入参数 2	<p>NewPostscaler: 定时器 Tz 的后分频器, 取值为 4 位数据, 范围如下:</p> <p>ATIM_POSTSCALER_DIV_1: 分频比为 1:1</p> <p>ATIM_POSTSCALER_DIV_2: 分频比为 1:2</p> <p>ATIM_POSTSCALER_DIV_3: 分频比为 1:3</p> <p>ATIM_POSTSCALER_DIV_4: 分频比为 1:4</p> <p>ATIM_POSTSCALER_DIV_5: 分频比为 1:5</p> <p>ATIM_POSTSCALER_DIV_6: 分频比为 1:6</p> <p>ATIM_POSTSCALER_DIV_7: 分频比为 1:7</p> <p>ATIM_POSTSCALER_DIV_8: 分频比为 1:8</p> <p>ATIM_POSTSCALER_DIV_9: 分频比为 1:9</p> <p>ATIM_POSTSCALER_DIV_10: 分频比为 1:10</p> <p>ATIM_POSTSCALER_DIV_11: 分频比为 1:11</p> <p>ATIM_POSTSCALER_DIV_12: 分频比为 1:12</p> <p>ATIM_POSTSCALER_DIV_13: 分频比为 1:13</p> <p>ATIM_POSTSCALER_DIV_14: 分频比为 1:14</p> <p>ATIM_POSTSCALER_DIV_15: 分频比为 1:15</p> <p>ATIM_POSTSCALER_DIV_16: 分频比为 1:16</p>
返回值	无
被调用函数	无

3. 4. 22 函数 ATIM_X_External_Pulse_Sync_Config

表 3-28 函数 ATIM_X_External_Pulse_Sync_Config

函数名	ATIM_X_External_Pulse_Sync_Config
函数原型	void ATIM_X_External_Pulse_Sync_Config (ATIM_SFRmap* ATIMx, uint32_t PulseSync)
功能描述	配置定时器 Tx 计数模式外部触发脉冲输入同步控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	<p>NewClock: 定时器 Tx 计数模式外部触发脉冲输入同步控制位使能状态, 取值范围为:</p> <p>ATIM_EX_SYNC_MODE: 与外部触发脉冲输入同步</p> <p>ATIM_NO_SYNC_MODE: 不与外部触发脉冲输入同步</p>
返回值	无
被调用函数	无

3. 4. 23 函数 ATIM_Z_External_Pulse_Sync_Config

表 3-29 函数 ATIM_Z_External_Pulse_Sync_Config

函数名	ATIM_Z_External_Pulse_Sync_Config
函数原型	void ATIM_Z_External_Pulse_Sync_Config (ATIM_SFRmap* ATIMx, uint32_t PulseSync)

功能描述	配置定时器 Tz 计数模式外部触发脉冲输入同步控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewClock: 定时器 Tz 计数模式外部触发脉冲输入同步控制位使能状态, 取值范围为: ATIM_EX_SYNC_MODE: 与外部触发脉冲输入同步 ATIM_NO_SYNC_MODE: 不与外部触发脉冲输入同步
返回值	无
被调用函数	无

3. 4. 24 函数 ATIM_X_Work_Mode_Config

表 3-30 函数 ATIM_X_Work_Mode_Config

函数名	ATIM_X_Work_Mode_Config
函数原型	void ATIM_X_Work_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t NewState)
功能描述	配置 Tx 定时/计数模式选择
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 定时/计数模式选择, 取值范围为: ATIM_TIMER_MODE: 定时模式 ATIM_COUNTER_MODE: 计数模式
返回值	无
被调用函数	无

3. 4. 25 函数 ATIM_Z_Work_Mode_Config

表 3-31 函数 ATIM_Z_Work_Mode_Config

函数名	ATIM_Z_Work_Mode_Config
函数原型	void ATIM_Z_Work_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t NewState)
功能描述	配置 Tz 定时/计数模式选择
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewState: Tz 定时/计数模式选择, 取值范围为: ATIM_TIMER_MODE: 定时模式 ATIM_COUNTER_MODE: 计数模式
返回值	无
被调用函数	无

3.4.26 函数 ATIM_X_Get_Direction

表 3-32 函数 ATIM_X_Get_Direction

函数名	ATIM_X_Get_Direction
函数原型	DIRStatus ATIM_X_Get_Direction (ATIM_SFRmap* ATIMx)
功能描述	读 Tx 计数方向
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	Tx 计数方向, 0: 向下, 1: 向上
被调用函数	无

3.4.27 函数 ATIM_Z_Get_Direction

表 3-33 函数 ATIM_Z_Get_Direction

函数名	ATIM_Z_Get_Direction
函数原型	DIRStatus ATIM_Z_Get_Direction (ATIM_SFRmap* ATIMx)
功能描述	读 Tz 计数方向
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	Tz 计数方向, 0: 向下, 1: 向上
被调用函数	无

3.4.28 函数 ATIM_X_Overflow_AD_Enable

表 3-34 函数 ATIM_X_Overflow_AD_Enable

函数名	ATIM_X_Overflow_AD_Enable
函数原型	void ATIM_X_Overflow_AD_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 上溢中断触发 AD 使能位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 上溢中断触发 AD 使能位, 取值为: TRUE 或 FALSE。
7 返回值	无
被调用函数	无

3.4.29 函数 ATIM_Z_Overflow_AD_Enable

表 3-35 函数 ATIM_Z_Overflow_AD_Enable

函数名	ATIM_Z_Overflow_AD_Enable
函数原型	void ATIM_Z_Overflow_AD_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tz 上溢中断触发 AD 使能位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR

输入参数 2	NewState: Tz 上溢中断触发 AD 使能位，取值为：TRUE 或 FALSE。
7 返回值	无
被调用函数	无

3. 4. 30 函数 ATIM_X_Underflow_AD_Enable

表 3-36 函数 ATIM_X_Underflow_AD_Enable

函数名	ATIM_X_Underflow_AD_Enable
函数原型	void ATIM_X_Underflow_AD_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 下溢中断触发 AD 使能位
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 下溢中断触发 AD 使能位，取值为：TRUE 或 FALSE。
返回值	无
被调用函数	无

3. 4. 31 函数 ATIM_Z_Underflow_AD_Enable

表 3-37 函数 ATIM_Z_Underflow_AD_Enable

函数名	ATIM_Z_Underflow_AD_Enable
函数原型	void ATIM_Z_Underflow_AD_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tz 下溢中断触发 AD 使能位
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T6_SFR/T10_SFR
输入参数 2	NewState: Tz 下溢中断触发 AD 使能位，取值为：TRUE 或 FALSE。
返回值	无
被调用函数	无

3. 4. 32 函数 ATIM_X_TriggerAD_Config

表 3-38 函数 ATIM_X_TriggerAD_Config

函数名	ATIM_X_TriggerAD_Config
函数原型	void ATIM_X_TriggerAD_Config (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 溢出中断自动触发 AD 控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 溢出中断自动触发 AD 控制位，取值为：TRUE 或 FALSE。
返回值	无
被调用函数	无

3.4.33 函数 ATIM_Z_TriggerAD_Config

表 3-39 函数 ATIM_Z_TriggerAD_Config

函数名	ATIM_Z_TriggerAD_Config
函数原型	void ATIM_Z_TriggerAD_Config (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tz 溢出中断自动触发 AD 控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewState: Tz 溢出中断自动触发 AD 控制位, 取值为: TRUE 或 FALSE。
返回值	无
被调用函数	无

3.4.34 函数 ATIM_X_Set_TriggerAD_Signal

表 3-40 函数 ATIM_X_Set_TriggerAD_Signal

函数名	ATIM_X_Set_TriggerAD_Signal
函数原型	void ATIM_X_Set_TriggerAD_Signal (ATIM_SFRmap* ATIMx, uint16_t CompareAD0, uint16_t CompareAD1)
功能描述	配置定时器 Tx 触发 AD 寄存器的值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	CompareAD0: Tx 触发 AD 寄存器 0 的值, 取值 16 位数据
输入参数 3	CompareAD1: Tx 触发 AD 寄存器 1 的值, 取值 16 位数据
返回值	无
被调用函数	无

3.4.35 函数 ATIM_Z_Set_TriggerAD_Signal

表 3-41 函数 ATIM_Z_Set_TriggerAD_Signal

函数名	ATIM_Z_Set_TriggerAD_Signal
函数原型	void ATIM_Z_Set_TriggerAD_Signal (ATIM_SFRmap* ATIMx, uint16_t CompareAD0, uint16_t CompareAD1)
功能描述	配置定时器 Tz 触发 AD 寄存器的值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	CompareAD0: Tz 触发 AD 寄存器 0 的值, 取值 16 位数据
输入参数 3	CompareAD1: Tz 触发 AD 寄存器 1 的值, 取值 16 位数据
返回值	无
被调用函数	无

3. 4. 36 函数 ATIM_X_Update_Immediately_Config

表 3-42 函数 ATIM_X_Update_Immediately_Config

函数名	ATIM_X_Update_Immediately_Config
函数原型	void ATIM_X_Update_Immediately_Config (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 立即更新控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: 立即更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 37 函数 ATIM_Z_Update_Immediately_Config

表 3-43 函数 ATIM_Z_Update_Immediately_Config

函数名	ATIM_Z_Update_Immediately_Config
函数原型	void ATIM_Z_Update_Immediately_Config (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tz 立即更新控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewState: 立即更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 38 函数 ATIM_X_Update_Output_Ctl

表 3-44 函数 ATIM_X_Update_Output_Ctl

函数名	ATIM_X_Update_Output_Ctl
函数原型	void ATIM_X_Update_Output_Ctl (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 输出控制寄存器更新控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: 立即更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 39 函数 ATIM_Z_Update_Output_Ctl

表 3-45 函数 ATIM_Z_Update_Output_Ctl

函数名	ATIM_Z_Update_Output_Ctl
-----	--------------------------

函数原型	void ATIM_Z_Updata_Output_Ctl (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	配置 Tz 输出控制寄存器更新控制位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewState: 立即更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 40 函数 ATIM_X_Updata_Enable

表 3-46 函数 ATIM_X_Updata_Enable

函数名	ATIM_X_Updata_Enable
函数原型	void ATIM_Z_Updata_Output_Ctl (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	高级定时器(ATIM)配置, 配置 Tz
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: 更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 41 函数 ATIM_Z_Updata_Enable

表 3-47 函数 ATIM_Z_Updata_Enable

函数名	ATIM_Z_Updata_Enable
函数原型	void ATIM_Z_Updata_Output_Ctl (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	高级定时器(ATIM)配置, 配置 Tz
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	NewState: 更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 42 函数 ATIM_X_Set_Updata_Counter

表 3-48 函数 ATIM_X_Set_Updata_Counter

函数名	ATIM_X_Set_Updata_Counter
函数原型	void ATIM_X_Set_Updata_Counter (ATIM_SFRmap* ATIMx, uint8_t UpdataCounter)
功能描述	配置定时器 Tx 更新计数器
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR

输入参数 2	UpdateCounter: 更新计数值, 取值 8 位数据
返回值	无
被调用函数	无

3. 4. 43 函数 ATIM_Z_Set_Update_Counter

表 3-49 函数 ATIM_Z_Set_Update_Counter

函数名	ATIM_Z_Set_Update_Counter
函数原型	void ATIM_Z_Set_Update_Counter (ATIM_SFRmap* ATIMx, uint8_t UpdateCounter)
功能描述	配置定时器 Tz 更新计数器
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	UpdateCounter: 更新计数值, 取值 8 位数据
返回值	无
被调用函数	无

3. 4. 44 函数 ATIM_X_Slave_Mode_Config

表 3-50 函数 ATIM_X_Slave_Mode_Config

函数名	ATIM_X_Slave_Mode_Config
函数原型	void ATIM_X_Slave_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t SlaveMode)
功能描述	配置 Tx 从模式选择位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	SlaveSelect: 从模式选择, 取值范围为: ATIM_SLAVE_FORBIDDEN_MODE: 从模式禁止 ATIM_SLAVE_TRIGGER_MODE: 触发模式 ATIM_SLAVE_GATED_MODE: 门控模式 ATIM_SLAVE_RESET_MODE: 复位模式 ATIM_SLAVE_COUNTER_MODE: 计数模式 2
返回值	无
被调用函数	无

3. 4. 45 函数 ATIM_Z_Slave_Mode_Config

表 3-51 函数 ATIM_Z_Slave_Mode_Config

函数名	ATIM_Z_Slave_Mode_Config
函数原型	void ATIM_Z_Slave_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t SlaveMode)
功能描述	配置 Tz 从模式选择位

输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	SlaveSelect: 从模式选择, 取值范围为: ATIM_SLAVE_FORBIDDEN_MODE: 从模式禁止 ATIM_SLAVE_TRIGGER_MODE: 触发模式 ATIM_SLAVE_GATED_MODE: 门控模式 ATIM_SLAVE_RESET_MODE: 复位模式 ATIM_SLAVE_COUNTER_MODE: 计数模式 2
返回值	无
被调用函数	无

3.4.46 函数 ATIM_Master_Mode_Config

表 3-52 函数 ATIM_Master_Mode_Config

函数名	ATIM_Master_Mode_Config
函数原型	void ATIM_Master_Mode_Config (ATIM_SFRmap* ATIMx, uint32_t MasterMode)
功能描述	配置 Tx 主模式选择位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	MasterMode: 主模式选择, 取值范围为: ATIM_MASTER_TXEN_SIGNAL: 使能信号 TXEN 作为触发输出 TRGO ATIM_MASTER_TXIF_SIGNAL: TXIF 信号作为触发输出 TRGO (定时器溢出时的中断脉冲信号) ATIM_MASTER_CCPXCH1IF_SIGNAL: ECCPx 的 CC1IF 脉冲作为触发输出 TRGO ATIM_MASTER_CCPXCH1_SIGNAL: ECCPxCH1 作为触发输出 TRGO ATIM_MASTER_CCPXCH2_SIGNAL: ECCPxCH2 作为触发输出 TRGO ATIM_MASTER_CCPXCH3_SIGNAL: ECCPxCH3 作为触发输出 TRGO ATIM_MASTER_CCPXCH4_SIGNAL: ECCPxCH4 作为触发输出 TRGO
返回值	无
被调用函数	无

3.4.47 函数 ATIM_Master_Slave_Snyc_Enable

表 3-53 函数 ATIM_Master_Slave_Snyc_Enable

函数名	ATIM_Master_Slave_Snyc_Enable
函数原型	void ATIM_Master_Slave_Snyc_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置主从模式同步位
输入参数 1	ATIMx:指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState:主从模式同步位状态, 取值范围为: TRUE 或 FALSE
返回值	无

被调用函数	无
-------	---

3. 4. 48 函数 ATIM_Trigger_Select_Config

表 3-54 函数 ATIM_Trigger_Select_Config

函数名	ATIM_Trigger_Select_Config
函数原型	void ATIM_Trigger_Select_Config (ATIM_SFRmap* ATIMx,uint32_t TriggerSelect)
功能描述	配置触发选择位
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	TriggerSelect: 触发选择, 取值范围为: ATIM_TRIGGER_T1 ATIM_TRIGGER_T20 ATIM_TRIGGER_T5 ATIM_TRIGGER_T9 ATIM_TRIGGER_ECCPXCH1 ATIM_TRIGGER_ECCPXCH2 ATIM_TRIGGER_ECCPXCH3 ATIM_TRIGGER_TXCK
返回值	无
被调用函数	无

3. 4. 49 函数 ATIM_Timer_Unite_Enable

表 3-55 函数 ATIM_Timer_Unite_Enable

函数名	ATIM_Timer_Unite_Enable
函数原型	void ATIM_Timer_Unite_Enable (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	配置 Tx、Tz 联立使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx、Tz 联立使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 50 函数 ATIM_X_Get_Counter

表 3-56 函数 ATIM_X_Get_Counter

函数名	ATIM_X_Get_Counter
函数原型	void ATIM_Timer_Unite_Enable (ATIM_SFRmap* ATIMx,FunctionalState NewState)

功能描述	读 Tx 定时器计数值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	Tx 计数值, 16 位
被调用函数	无

3.4.51 函数 ATIM_Z_Get_Counter

表 3-57 函数 ATIM_Z_Get_Counter

函数名	ATIM_Z_Get_Counter
函数原型	void ATIM_Timer_Unite_Enable (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	读 Tz 定时器计数值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	Tz 计数值, 16 位
被调用函数	无

3.4.52 函数 ATIM_X_Get_Period

表 3-58 函数 ATIM_X_Get_Period

函数名	ATIM_X_Get_Period
函数原型	uint16_t ATIM_X_Get_Period (ATIM_SFRmap* ATIMx)
功能描述	读 Tx 定时器周期值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	Tx 周期值, 16 位
被调用函数	无

3.4.53 函数 ATIM_Z_Get_Period

表 3-59 函数 ATIM_Z_Get_Period

函数名	ATIM_Z_Get_Period
函数原型	uint16_t ATIM_Z_Get_Period (ATIM_SFRmap* ATIMx)
功能描述	读 Tz 定时器周期值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	Tz 周期值, 16 位
被调用函数	无

3.4.54 函数 ATIM_X_Get_Prescaler

表 3-60 函数 ATIM_X_Get_Prescaler

函数名	ATIM_X_Get_Prescaler
函数原型	uint16_t ATIM_X_Get_Prescaler (ATIM_SFRmap* ATIMx)
功能描述	读 Tx 定时器预分频值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	Tx 预分频值, 16 位
被调用函数	无

3.4.55 函数 ATIM_Z_Get_Prescaler

表 3-61 函数 ATIM_Z_Get_Prescaler

函数名	ATIM_Z_Get_Prescaler
函数原型	uint16_t ATIM_Z_Get_Prescaler (ATIM_SFRmap* ATIMx)
功能描述	读 Tz 定时器预分频值
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	Tz 预分频值, 16 位
被调用函数	无

3.4.56 函数 ATIM_X_Udata_INT_Enable

表 3-62 函数 ATIM_X_Udata_INT_Enable

函数名	ATIM_X_Udata_INT_Enable
函数原型	void ATIM_X_Udata_INT_Enable (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	配置 Tx 更新事件中断使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 更新事件中断, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.57 函数 ATIM_Z_Udata_INT_Enable

表 3-63 函数 ATIM_Z_Udata_INT_Enable

函数名	ATIM_Z_Udata_INT_Enable
函数原型	void ATIM_Z_Udata_INT_Enable (ATIM_SFRmap* ATIMx,FunctionalState NewState)
功能描述	配置 Tz 更新事件中断使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR

输入参数 2	NewState: Tz 更新事件中断，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 58 函数 ATIM_X_Overflow_INT_Enable

表 3-64 函数 ATIM_X_Overflow_INT_Enable

函数名	ATIM_X_Overflow_INT_Enable
函数原型	void ATIM_X_Overflow_INT_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 计数溢出中断使能
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 计数溢出中断，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 59 函数 ATIM_Z_Overflow_INT_Enable

表 3-65 函数 ATIM_Z_Overflow_INT_Enable

函数名	ATIM_Z_Overflow_INT_Enable
函数原型	void ATIM_Z_Overflow_INT_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tz 计数溢出中断使能
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tz 计数溢出中断，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 60 函数 ATIM_X_Trigger_INT_Enable

表 3-66 函数 ATIM_X_Trigger_INT_Enable

函数名	ATIM_X_Trigger_INT_Enable
函数原型	void ATIM_X_Trigger_INT_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 触发事件中断使能
输入参数 1	ATIMx: 指向定时器内存结构的指针，取值 T5_SFR/T9_SFR
输入参数 2	NewState: Tx 触发事件中断，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 61 函数 ATIM_X_Updata_DMA_Enable

表 3-67 函数 ATIM_X_Updata_DMA_Enable

函数名	ATIM_X_Updata_DMA_Enable
函数原型	void ATIM_X_Updata_DMA_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 Tx 更新事件的 DMA 请求使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 ECCP5_SFR/ECCP9_SFR
输入参数 2	NewState: Tx 更新事件的 DMA 请求, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 62 函数 ATIM_Z_Updata_DMA_Enable

表 3-68 函数 ATIM_Z_Updata_DMA_Enable

函数名	ATIM_Z_Updata_DMA_Enable
函数原型	void ATIM_Z_Updata_DMA_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	配置 _Z_ 更新事件的 DMA 请求使能
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 ECCP6_SFR/ECCP10_SFR
输入参数 2	NewState: _Z_ 更新事件的 DMA 请求, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 63 函数 ATIM_X_Trigger_DMA_Enable

表 3-69 函数 ATIM_X_Trigger_DMA_Enable

函数名	ATIM_X_Trigger_DMA_Enable
函数原型	void ATIM_X_Trigger_DMA_Enable (ATIM_SFRmap* ATIMx, FunctionalState NewState)
功能描述	允许 Tx 触发事件的 DMA 请求配置
输入参数 1	ECCPx:指向 CCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR /ECCP9_SFR
输入参数 2	NewState: Tx 触发事件的 DMA 请求使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.64 函数 ATIM_X_Get_Update_INT_Flag

表 3-70 函数 ATIM_X_Get_Update_INT_Flag

函数名	ATIM_X_Get_Update_INT_Flag
函数原型	FlagStatus ATIM_X_Get_Update_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tx 更新事件中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.65 函数 ATIM_Z_Get_Update_INT_Flag

表 3-71 函数 ATIM_Z_Get_Update_INT_Flag

函数名	ATIM_Z_Get_Update_INT_Flag
函数原型	FlagStatus ATIM_Z_Get_Update_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tz 更新事件中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.66 函数 ATIM_X_Get_Overflow_INT_Flag

表 3-72 函数 ATIM_X_Get_Overflow_INT_Flag

函数名	ATIM_X_Get_Overflow_INT_Flag
函数原型	FlagStatus ATIM_Z_Get_Overflow_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tx 计数溢出中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.67 函数 ATIM_Z_Get_Overflow_INT_Flag

表 3-73 函数 ATIM_Z_Get_Overflow_INT_Flag

函数名	ATIM_Z_Get_Overflow_INT_Flag
函数原型	FlagStatus ATIM_Z_Get_Overflow_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tz 计数溢出中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.68 函数 ATIM_X_Get_Trigger_INT_Flag

表 3-74 函数 ATIM_X_Get_Trigger_INT_Flag

函数名	ATIM_X_Get_Trigger_INT_Flag
函数原型	FlagStatus ATIM_X_Get_Trigger_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tx 触发事件中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.69 函数 ATIM_X_Generate_Trigger_Config

表 3-75 函数 ATIM_X_Generate_Trigger_Config

函数名	ATIM_X_Generate_Trigger_Config
函数原型	void ECCP_Generate_Trigger_Config (ECCP_SFRmap* ECCPx, uint32_t Channel, FunctionalState NewState)
功能描述	产生捕捉/比较事件配置位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR /ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	NewState: 产生捕捉/比较事件配置状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.70 函数 ATIM_X_Get_Updata_DMA_INT_Flag

表 3-76 函数 ATIM_X_Get_Updata_DMA_INT_Flag

函数名	ATIM_X_Get_Updata_DMA_INT_Flag
函数原型	FlagStatus ATIM_X_Get_Updata_DMA_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tx 更新事件 DMA 中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.71 函数 ATIM_Z_Get_Updata_DMA_INT_Flag

表 3-77 函数 ATIM_Z_Get_Updata_DMA_INT_Flag

函数名	ATIM_Z_Get_Updata_DMA_INT_Flag
函数原型	FlagStatus ATIM_Z_Get_Updata_DMA_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tz 更新事件 DMA 中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.72 函数 ATIM_X_Get_Trigger_DMA_INT_Flag

表 3-78 函数 ATIM_X_Get_Trigger_DMA_INT_Flag

函数名	ATIM_X_Get_Trigger_DMA_INT_Flag
函数原型	FlagStatus ATIM_X_Get_Trigger_DMA_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取 Tx 触发事件触发 DMA 中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.73 函数 ATIM_X_Clear_Updata_INT_Flag

表 3-79 函数 ATIM_X_Clear_Updata_INT_Flag

函数名	ATIM_X_Clear_Updata_INT_Flag
函数原型	void ATIM_X_Clear_Updata_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	清除 Tx 更新事件中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
输入参数 2	无; 特殊说明: 清除该标志位需在使能对应定时器的情况下进行, 否则会导致清除失败
返回值	无
被调用函数	无

3.4.74 函数 ATIM_Z_Clear_Updata_INT_Flag

表 3-80 函数 ATIM_Z_Clear_Updata_INT_Flag

函数名	ATIM_Z_Clear_Updata_INT_Flag
函数原型	void ATIM_Z_Clear_Updata_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	清除 Tz 更新事件中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
输入参数 2	无; 特殊说明: 清除该标志位需在使能对应定时器的情况下进行, 否则会导致清除失败

	导致清除失败
返回值	无
被调用函数	无

3.4.75 函数 ATIM_X_Clear_Overflow_INT_Flag

表 3-81 函数 ATIM_X_Clear_Overflow_INT_Flag

函数名	ATIM_X_Clear_Overflow_INT_Flag
函数原型	void ATIM_X_Clear_Overflow_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	清除 Tx 溢出中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	无
被调用函数	无

3.4.76 函数 ATIM_Z_Clear_Overflow_INT_Flag

表 3-82 函数 ATIM_Z_Clear_Overflow_INT_Flag

函数名	ATIM_Z_Clear_Overflow_INT_Flag
函数原型	void ATIM_Z_Clear_Overflow_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	清除 Tz 溢出中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T6_SFR/T10_SFR
返回值	无
被调用函数	无

3.4.77 函数 ATIM_X_Clear_Trigger_INT_Flag

表 3-83 函数 ATIM_X_Clear_Trigger_INT_Flag

函数名	ATIM_X_Clear_Trigger_INT_Flag
函数原型	void ATIM_X_Clear_Trigger_INT_Flag (ATIM_SFRmap* ATIMx)
功能描述	清除 Tx 触发事件中断标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	无
被调用函数	无

3.4.78 函数 ECCP_Compare_Configuration

表 3-84 函数 ECCP_Compare_Configuration

函数名	ECCP_Compare_Configuration
函数原型	void ECCP_Compare_Configuration(ECCP_SFRmap* ECCPx, uint32_t

	Channel,uint32_t CompareMode, uint16_t CompareValue)
功能描述	增强型比较器(ECCP)配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 比较通道, 取值为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	CompareMode: 比较器模式, 取值为: ECCP_MODE_RST: 关闭比较功能 ECCP_CMP_TOGGLE_LEVEL: 比较匹配时输出翻转 ECCP_CMP_ACTIVE_LEVEL: 比较匹配时输出高电平 ECCP_CMP_INACTIVE_LEVEL: 比较匹配时输出低电平 ECCP_CMP_SOFT_INT: 比较匹配时产生软中断 ECCP_CMP_SPECIAL_EVENT: 比较匹配时除法特殊事件
输入参数 4	CompareValue: 比较寄存器的值, 取值为 16 位数据
返回值	无
被调用函数	无

3. 4. 79 函数 ECCP_Capture_Configuration

表 3-85 函数 ECCP_Capture_Configuration

函数名	ECCP_Capture_Configuration
函数原型	void ECCP_Capture_Configuration (ECCP_SFRmap* ECCPx, ECCP_CaptureInitTypeDef* eccpInitStruct)
功能描述	增强型捕捉模块(ECCP)配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	eccpInitStruct: 增强型捕捉模块(ECCP)配置信息结构体指针
返回值	无
被调用函数	无

3. 4. 80 函数 ECCP_Capture_Struct_Init

表 3-86 函数 ECCP_Capture_Struct_Init

函数名	ECCP_Capture_Struct_Init
函数原型	void ECCP_Capture_Struct_Init (ECCP_CaptureInitTypeDef* eccpInitStruct)
功能描述	初始化增强型捕捉模块(ECCP)配置信息结构体
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/

	ECCP9_SFR
输入参数 2	eccpInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

3.4.81 函数 ECCP_PWM_Configuration

表 3-87 函数 ECCP_PWM_Configuration

函数名	ECCP_PWM_Configuration
函数原型	void ECCP_PWM_Configuration (ECCP_SFRmap* ECCPx, ECCP_PWMInitTypeDef * eccpInitStruct)
功能描述	增强型 PWM 模块(ECCP)配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	eccpInitStruct: 增强型 PWM 模块(ECCP)配置信息结构体指针
返回值	无
被调用函数	无

3.4.82 函数 ECCP_PWM_Struct_Init

表 3-88 函数 ECCP_PWM_Struct_Init

函数名	ECCP_PWM_Struct_Init
函数原型	void ECCP_PWM_Struct_Init (ECCP_PWMInitTypeDef* eccpInitStruct)
功能描述	初始化增强型 PWM 模块(ECCP)配置信息结构体
输入参数 1	eccpInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

3.4.83 函数 ECCP_Capture_Mode_Config

表 3-89 函数 ECCP_Capture_Mode_Config

函数名	ECCP_Capture_Mode_Config
函数原型	void ECCP_Capture_Mode_Config (ECCP_SFRmap* ECCPx,uint32_t Channel, uint32_t EdgeConfig)
功能描述	配置 ECCP 捕捉功能
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2

	ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	EdgeConfig: 匹配边沿, 取值范围为: ECCP_MODE_RST: 关闭捕捉功能 ECCP_CAP_RISING_EDGE: 每个下降沿发生捕捉 ECCP_CAP_FALLING_EDGE: 每个上升沿发生捕捉 ECCP_CAP_4TH_RISING_EDGE: 每 4 个上升沿发生捕捉 ECCP_CAP_16TH_RISING_EDGE: 每 16 个上升沿发生捕捉
返回值	无
被调用函数	无

3. 4. 84 函数 ECCP_Compare_Mode_Config

表 3-90 函数 ECCP_Compare_Mode_Config

函数名	ECCP_Compare_Mode_Config
函数原型	void ECCP_Compare_Mode_Config (ECCP_SFRmap* ECCPx,uint32_t Channel, uint32_t CmpConfig)
功能描述	配置 ECCP 比较功能
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 比较通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	CmpConfig: 比较匹配模式, 取值范围为: ECCP_MODE_RST: 关闭比较功能 ECCP_CMP_TOGGLE_LEVEL: 匹配时输出电平翻转 ECCP_CMP_ACTIVE_LEVEL: 比较匹配时输出高电平 ECCP_CMP_INACTIVE_LEVEL: 比较匹配时输出低电平 ECCP_CMP_SOFT_INT: 比较匹配时产生软件中断 ECCP_CMP_SPECIAL_EVENT: 触发特殊事件
返回值	无
被调用函数	无

3. 4. 85 函数 ECCP_PWM_Mode_Config

表 3-91 函数 ECCP_PWM_Mode_Config

函数名	ECCP_PWM_Mode_Config
函数原型	void ECCP_PWM_Mode_Config (ECCP_SFRmap* ECCPx, uint32_t PWMConfig)

功能描述	配置 ECCP PWM 功能,模式选择,自由,协同,单时机模式
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	PWMConfig: PWM 匹配模式, 取值范围为: ECCP_MODE_RST: 关闭比较功能 ECCP_PWM_FREE: PWM 自由模式 ECCP_PWM_COORDINATION: PWM 协同模式 ECCP_PWM_SINGLE: PWM 单时基模式
返回值	无
被调用函数	无

3.4.86 函数 ECCP_Get_Capture_Result

表 3-92 函数 ECCP_Get_Capture_Result

函数名	ECCP_Get_Capture_Result
函数原型	uint16_t ECCP_Get_Capture_Result (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	读取 ECCP 捕捉寄存器
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 T5_SFR/T9_SFR 或 ECCP5_SFR/ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	捕捉寄存器的 16 位值
被调用函数	无

3.4.87 函数 ECCP_Get_Compare_Result

表 3-93 函数 ECCP_Get_Compare_Result

函数名	ECCP_Get_Compare_Result
函数原型	void ECCP_PWM_Mode_Config (ECCP_SFRmap* ECCPx, uint32_t PWMConfig)
功能描述	读取 ECCP 比较/EPWM 占空比寄存器
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 T5_SFR/T9_SFR 或 ECCP5_SFR/ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3

	ECCP_CHANNEL_4: 通道 4
返回值	比较/PWM 占空比寄存器的值
被调用函数	无

3. 4. 88 函数 ECCP_Set_Compare_Result

表 3-94 函数 ECCP_Set_Compare_Result

函数名	ECCP_Set_Compare_Result
函数原型	void ECCP_Set_Compare_Result (ECCP_SFRmap* ECCPx, uint32_t Channel, uint16_t Value)
功能描述	配置 ECCP 比较/PWM 占空比寄存器
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	无
被调用函数	无

3. 4. 89 函数 ECCP_Generate_Trigger_Config

表 3-95 函数 ECCP_Generate_Trigger_Config

函数名	ECCP_Generate_Trigger_Config
函数原型	void ECCP_Generate_Trigger_Config (ECCP_SFRmap* ECCPx, uint32_t Channel, FunctionalState NewState)
功能描述	产生捕捉/比较事件配置位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	NewState: 产生捕捉/比较事件配置状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 90 函数 ECCP_PWM_Input_Enable

表 3-96 函数 ECCP_PWM_Input_Enable

函数名	ECCP_PWM_Input_Enable
函数原型	void ECCP_PWM_Input_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	PWM 输入测量模式使能
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: PWM 输入测量模式使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 91 函数 ECCP_Input_XOR_Enable

表 3-97 函数 ECCP_Input_XOR_Enable

函数名	ECCP_Input_XOR_Enable
函数原型	void ECCP_Input_XOR_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	输入异或使能位配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: PWM 输入异或使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 92 函数 ECCP_Single_Pulse_Enable

表 3-98 函数 ECCP_Single_Pulse_Enable

函数名	ECCP_Single_Pulse_Enable
函数原型	void ECCP_Single_Pulse_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	配置单脉冲输出模式
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: 单脉冲输出模式使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.93 函数 ECCP_Single_Pulse_Shut_Enable

表 3-99 函数 ECCP_Single_Pulse_Shut_Enable

函数名	ECCP_Single_Pulse_Shut_Enable
函数原型	void ECCP_Single_Pulse_Shut_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	配置单脉冲输出模式选择
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: 单脉冲输出模式选择, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.94 函数 ECCP_PWM_Restart_Enable

表 3-100 函数 ECCP_PWM_Restart_Enable

函数名	ECCP_PWM_Restart_Enable
函数原型	void ECCP_PWM_Restart_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	配置 PWM 通道 1/2/3/4 重启使能位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: PWM 通道 1/2/3/4 重启使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.95 函数 ECCP_Dead_Time_Config

表 3-101 函数 ECCP_Dead_Time_Config

函数名	ECCP_Dead_Time_Config
函数原型	void ECCP_Dead_Time_Config (ECCP_SFRmap* ECCPx, uint32_t Channel, uint8_t DeadTime)
功能描述	配置 PWM 通道死区延时时间
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4

输入参数 3	DeadTime: 死区延时时间, 取值为 8 位数据
返回值	无
被调用函数	无

3. 4. 96 函数 ECCP_Channel_Output_Control

表 3-102 函数 ECCP_Channel_Output_Control

函数名	ECCP_Channel_Output_Control
函数原型	void ECCP_Channel_Output_Control (ECCP_SFRmap* ECCPx, uint32_t Channel,uint32_t Port, uint32_t ChannelOutputCtl)
功能描述	配置 ECCP 通道输出控制
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	Port: 端口选择, 取值范围为: ECCP_PORT_LOW: CHxL 端口 ECCP_PORT_HIGH: CHxH 端口
输入参数 4	ChannelOutputCtl: 通道端口输出控制, 取值范围为: ECCP_CHANNEL_OUTPUT_PWM_ACTIVE: PWM 输出, 高有效 ECCP_CHANNEL_OUTPUT_PWM_INACTIVE: PWM 输出, 低有效 ECCP_CHANNEL_OUTPUT_INACTIVE: 强制低电平输出 ECCP_CHANNEL_OUTPUT_ACTIVE: 强制高电平输出
返回值	无
被调用函数	无

3. 4. 97 函数 ECCP_Channel_Output_Mode

表 3-103 函数 ECCP_Channel_Output_Mode

函数名	ECCP_Channel_Output_Mode
函数原型	void ECCP_Channel_Output_Mode (ECCP_SFRmap* ECCPx, uint32_t Channel,uint32_t ChannelOutputMode)
功能描述	配置 ECCP 通道输出模式
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1

	ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	ChannelOutputMode: PWM 通道输出模式, 取值范围为: ECCP_OUTPUT_INDEPENDENT: 独立输出模式 ECCP_OUTPUT_COMPLEMENTARY: 互补输出模式
返回值	无
被调用函数	无

3. 4. 98 函数 ECCP_Channel_Work_State_Config

表 3-104 函数 ECCP_Channel_Work_State_Config

函数名	ECCP_Channel_Work_State_Config
函数原型	void ECCP_Channel_Work_State_Config (ECCP_SFRmap* ECCPx, uint32_t Channel,uint32_t WorkingState)
功能描述	配置通道自动关闭事件状态位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	WorkingState: 通道自动关闭事件状态, 取值范围为: ECCP_CHANNEL_WORKING_STATE: 通道正常工作 ECCP_CHANNEL_SHUTDOWN_STATE: 通道关闭
返回值	无
被调用函数	无

3. 4. 99 函数 ECCP_Get_Channel_Work_State

表 3-105 函数 ECCP_Get_Channel_Work_State

函数名	ECCP_Get_Channel_Work_State
函数原型	FlagStatus ECCP_Get_Channel_Work_State (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	读取通道自动关闭事件状态位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1

	ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	通道自动关闭事件状态, 发生关闭事件返回 1, 正常工作返回 0
被调用函数	无

3. 4. 100 函数 ECCP_CHANNEL4_Shutdown_SEL

表 3-106 函数 ECCP_CHANNEL4_Shutdown_SEL

函数名	ECCP_CHANNEL4_Shutdown_SEL
函数原型	void ECCP_CHANNEL4_Shutdown_SEL (ECCP_SFRmap* ECCPx, uint32_t ShutDownSignal)
功能描述	配置通道自动关闭源选择位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	ShutDownSignal: 通道自动关闭源, 取值范围为: ECCP_CHANNEL_SHUTDOWN_FORBID: 禁止自动关断 ECCP_CHANNEL_CMP2CMP3_ACTIVE: 比较器 2/3 输出高电平 ECCP_CHANNEL_BKIN_INACTIVE: ECCP_BKIN 引脚上的低电平
注 1	注意: 比较器 3 输出作为 ECCP5/9 通道 1/2/3 自动关断的触发信号(高电平关断),比较器 2/3 的输出作为 ECCP5/9 通道 4 自动关断的触发信号(高电平关断)
返回值	无
被调用函数	无

3. 4. 101 函数 ECCP_Channel_Shutdown_Signal

表 3-107 函数 ECCP_Channel_Shutdown_Signal

函数名	ECCP_Channel_Shutdown_Signal
函数原型	void ECCP_Channel_Shutdown_Signal (ECCP_SFRmap* ECCPx, uint32_t Channel,uint32_t ShutDownSignal)
功能描述	配置通道自动关闭源选择位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为:

	ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	ShutDownSignal: 通道自动关闭源, 取值范围为: ECCP_CHANNEL_SHUTDOWN_FORBID: 禁止自动关断 ECCP_CHANNEL_CMP2CMP3_ACTIVE: 比较器 2/3 输出高电平 ECCP_CHANNEL_BKIN_INACTIVE: ECCP_BKIN 引脚上的低电平
注 1	注意: 比较器 3 输出作为 ECCP5/9 通道 1/2/3 自动关断的触发信号(高电平关断),比较器 2/3 的输出作为 ECCP5/9 通道 4 自动关断的触发信号(高电平关断)
返回值	无
被调用函数	无

3.4.102 函数 ECCP_Channel_Pin_Ctl

表 3-108 函数 ECCP_Channel_Pin_Ctl

函数名	ECCP_Channel_Pin_Ctl
函数原型	void ECCP_Channel_Pin_Ctl (ECCP_SFRmap* ECCPx, uint32_t Channel, uint32_t Port, uint32_t ChannelPinCtl)
功能描述	配置引脚 ECCPxCHyH/ECCPxCHyL 关闭状态控制位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	Port: 端口选择, 取值范围为: ECCP_PORT_LOW: CHxL 端口 ECCP_PORT_HIGH: CHxH 端口
输入参数 4	ChannelPinCtl: 引脚状态, 取值范围为: PIN_INACTIVE: 驱动引脚为低电平 PIN_ACTIVE: 驱动引脚为高电平 PIN_TRISTATE: 驱动引脚为三态
返回值	无
被调用函数	无

3.4.103 函数 ECCP_Zero_Clock_Config

表 3-109 函数 ECCP_Zero_Clock_Config

函数名	ECCP_Zero_Clock_Config
函数原型	void ECCP_Zero_Clock_Config (ECCP_SFRmap* ECCPx,uint32_t ZeroClock)
功能描述	配置零点检测时钟
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	ZeroClock: 零点检测时钟, 取值范围为: ECCP_ZERO_CLOCK_DIV_1: 预分频时钟 ECCP_ZERO_CLOCK_DIV_2: 预分频时钟 / 2 ECCP_ZERO_CLOCK_DIV_4: 预分频时钟 / 4 ECCP_ZERO_CLOCK_DIV_8: 预分频时钟 / 8 ECCP_ZERO_CLOCK_DIV_16: 预分频时钟 / 16 ECCP_ZERO_CLOCK_DIV_32: 预分频时钟 / 32 ECCP_ZERO_CLOCK_DIV_64: 预分频时钟 / 64 ECCP_ZERO_CLOCK_DIV_128: 预分频时钟 / 128 ECCP_ZERO_CLOCK_DIV_256: 预分频时钟 / 256 ECCP_ZERO_CLOCK_DIV_512: 预分频时钟 / 512
返回值	无
被调用函数	无

3.4.104 函数 ECCP_Channel_Pin_Tristate_Enable

表 3-110 函数 ECCP_Channel_Pin_Tristate_Enable

函数名	ECCP_Channel_Pin_Tristate_Enable
函数原型	void ECCP_Channel_Pin_Tristate_Enable (ECCP_SFRmap* ECCPx, uint32_t Channel,uint32_t Port, uint32_t PinTristateCtl)
功能描述	配置引脚 ECCPxCHyH/ECCPxCHyL 三态控制位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 捕捉通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	Port: 端口选择, 取值范围为: ECCP_PORT_LOW: CHxL 端口 ECCP_PORT_HIGH: CHxH 端口
输入参数 4	PinTristateCtl: 引脚状态, 取值范围为:

	PIN_INACTIVE: 驱动引脚为低电平 PIN_ACTIVE: 驱动引脚为高电平 PIN_TRISTATE: 驱动引脚为三态
返回值	无
被调用函数	无

3. 4. 105 函数 ECCP_Channel_INT_Enable

表 3-111 函数 ECCP_Channel_INT_Enable

函数名	ECCP_Channel_INT_Enable
函数原型	void ECCP_Channel_INT_Enable (ECCP_SFRmap* ECCPx, uint32_t Channel,FunctionalState NewState)
功能描述	通道捕获/比较的中断使能配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针，取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择，取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	NewState: 通道捕获/比较的中断使能状态，取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 106 函数 ECCP_X_Turn_off_DMA_Enable

表 3-112 函数 ECCP_X_Turn_off_DMA_Enable

函数名	ECCP_X_Turn_off_DMA_Enable
函数原型	void ECCP_X_Turn_off_DMA_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	关断事件的 DMA 请求使能位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针，取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: 关断事件的 DMA 请求使能状态,取值范围为:TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.107 函数 ECCP_Channel_DMA_Enable

表 3-113 函数 ECCP_Channel_DMA_Enable

函数名	ECCP_Channel_DMA_Enable
函数原型	void ECCP_Channel_DMA_Enable (ECCP_SFRmap* ECCPx, uint32_t Channel, FunctionalState NewState)
功能描述	通道捕获/比较的 DMA 请求允许使能位配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	NewState: 通道捕获/比较的 DMA 请求允许使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.108 函数 ECCP_Get_Channel_Trigger_INT_Flag

表 3-114 函数 ECCP_Get_Channel_Trigger_INT_Flag

函数名	ECCP_Get_Channel_Trigger_INT_Flag
函数原型	FlagStatus ECCP_Get_Channel_Trigger_INT_Flag (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	读取 ECCP 通道捕捉/比较中断标志
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3.4.109 函数 ECCP_X_Get_Turn_off_DMA_Flag

表 3-115 函数 ECCP_X_Get_Turn_off_DMA_Flag

函数名	ECCP_X_Get_Turn_off_DMA_Flag
-----	------------------------------

函数原型	FlagStatus ECCP_X_Get_Turn_off_DMA_Flag (ATIM_SFRmap* ATIMx)
功能描述	读取关断事件触发 DMA 标志
输入参数 1	ATIMx: 指向定时器内存结构的指针, 取值 T5_SFR/T9_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

3. 4. 110 函数 ECCP_Get_Trigger_DMA_INT_Flag

表 3-116 函数 ECCP_Get_Trigger_DMA_INT_Flag

函数名	ECCP_Get_Trigger_DMA_INT_Flag
函数原型	FlagStatus ECCP_Get_Trigger_DMA_INT_Flag (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	读取 ECCP 通道捕捉/比较 DMA 中断标志
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	无
被调用函数	无

3. 4. 111 函数 ECCP_Clear_Channel_INT_Flag

表 3-117 函数 ECCP_Clear_Channel_INT_Flag

函数名	ECCP_Clear_Channel_INT_Flag
函数原型	void ECCP_Clear_Channel_INT_Flag (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	清除 ECCPx 通道捕捉/比较中断标志
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	无
被调用函数	无

3. 4. 112 函数 ECCP_PWM_Move_Phase_Enable

表 3-118 函数 ECCP_PWM_Move_Phase_Enable

函数名	ECCP_PWM_Move_Phase_Enable
函数原型	void ECCP_PWM_Move_Phase_Enable (ECCP_SFRmap* ECCPx, FunctionalState NewState)
功能描述	PWM 相位移动使能位配置
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	NewState: PWM 相位移动使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 113 函数 ECCP_Channel_Zero_Detect_Sequential_Ctl

表 3-119 函数 ECCP_Channel_Zero_Detect_Sequential_Ctl

函数名	ECCP_Channel_Zero_Detect_Sequential_Ctl
函数原型	void ECCP_Channel_Zero_Detect_Sequential_Ctl (ECCP_SFRmap* ECCPx, uint32_t Channel, FunctionalState NewState)
功能描述	配置通道的零点检测时序控制
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	NewState: 通道捕获/比较的中断使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3. 4. 114 函数 ECCP_Get_Channel_Zero_Detection_State

表 3-120 函数 ECCP_Get_Channel_Zero_Detection_State

函数名	ECCP_Get_Channel_Zero_Detection_State
函数原型	FlagStatus ECCP_Get_Channel_Zero_Detection_State (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	读取通道零点检测的感应电压检测状态
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR

输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	返回 通道零点检测的感应电压检测状态, 未检测到感应电压 (检测到 0 点) 返回 1, 检测到感应电压 (未检测到 0 点) 返回 0。
被调用函数	无

3. 4. 115 函数 ECCP_Clear_Channel_Zero_Detection_State

表 3-121 函数 ECCP_Clear_Channel_Zero_Detection_State

函数名	ECCP_Clear_Channel_Zero_Detection_State
函数原型	void ECCP_Clear_Channel_Zero_Detection_State (ECCP_SFRmap* ECCPx, uint32_t Channel)
功能描述	清零通道零点检测的感应电压检测状态
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
返回值	无
被调用函数	无

3. 4. 116 函数 ECCP_Channel_Zero_Detect_Enable

表 3-122 函数 ECCP_Channel_Zero_Detect_Enable

函数名	ECCP_Channel_Zero_Detect_Enable
函数原型	void ECCP_Channel_Zero_Detect_Enable (ECCP_SFRmap* ECCPx, uint32_t Channel, FunctionalState NewState)
功能描述	配置通道的零点检测比较器使能位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3

	ECCP_CHANNEL_4: 通道 4
输入参数 3	NewState: 通道捕获/比较的中断使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

3.4.117 函数 ECCP_Channel_Zero_Voltage_Config

表 3-123 函数 ECCP_Channel_Zero_Voltage_Config

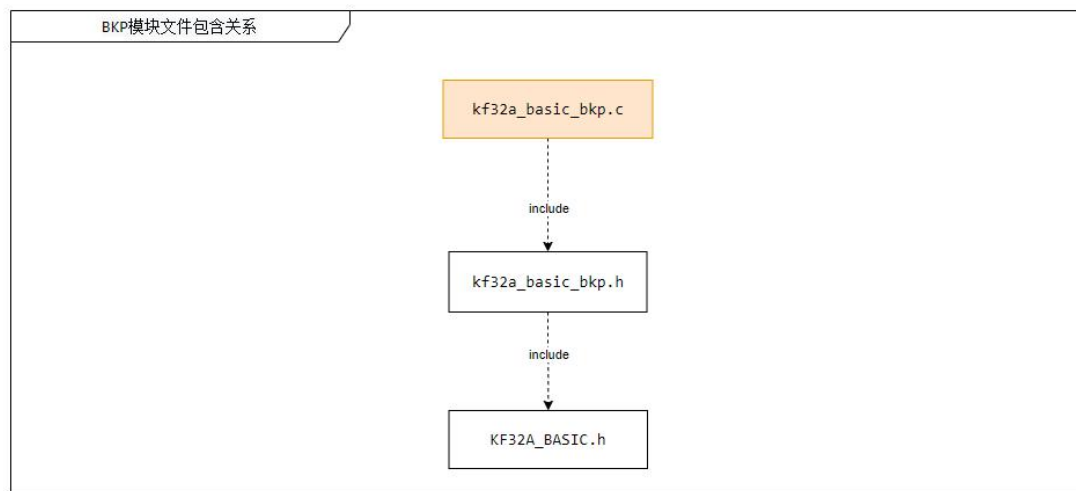
函数名	ECCP_Channel_Zero_Voltage_Config
函数原型	void ECCP_Channel_Zero_Voltage_Config (ECCP_SFRmap* ECCPx, uint32_t Channel, uint32_t ZeroDetectVoltage)
功能描述	配置通道零点检测电压选择位
输入参数 1	ECCPx: 指向 ECCP 或高级定时器内存结构的指针, 取值 ECCP5_SFR/ ECCP9_SFR
输入参数 2	Channel: 通道选择, 取值范围为: ECCP_CHANNEL_1: 通道 1 ECCP_CHANNEL_2: 通道 2 ECCP_CHANNEL_3: 通道 3 ECCP_CHANNEL_4: 通道 4
输入参数 3	ZeroDetectVoltage: 通道零点检测电压, 取值范围为: ECCP_ZERO_VOLTAGE_DECIMAL_15: 0.15V ECCP_ZERO_VOLTAGE_DECIMAL_25: 0.25V ECCP_ZERO_VOLTAGE_DECIMAL_35: 0.35V ECCP_ZERO_VOLTAGE_DECIMAL_45: 0.45V ECCP_ZERO_VOLTAGE_DECIMAL_55: 0.55V
返回值	无
被调用函数	无

4 备份域（BKP）

备份域（BKP）带有侵入检测功能的备份寄存器，可用于保存数据。备份域内寄存器只会在初始上电复位时被复位，不会因为 VDD 掉电上电而复位。

4.1 文件引用关系

图 4-1 BKP 模块文件包含关系



4.2 BKP 寄存器结构

BKP 寄存器结构，BKP_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct BKP_MemMap
{
    volatile uint32_t    CTL;
    volatile uint32_t    INT;
    uint32_t    RESERVED[14];
    volatile uint32_t    DATA[8];
}BKP_SFRmap;
  
```

表 4-1 BKP 寄存器结构说明

寄存器	描述
CTL	备份域控制寄存器, 偏移:0x00
INT	备份域中断控制寄存器, 偏移:0x04
RESERVED[14]	保留地址, 偏移:0x08
DATA[8]	备份域数据寄存器(0~7), 偏移:0x40

4.3 BKP 宏定义

BKP 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, BKP 其他相关宏定义详见文件 kf32a_basic_bkp.h 及函数参数描述。

4.4 BKP 库函数

表 4-2 BKP 固件库函数列表

序号	函数名	描述
1	BKP_Reset	复位备份域(BKP)外设, 该函数仅复位备份域寄存器组, 而不复位 RTC 模块
2	BKP_Write_And_Read_Enable	备份域(BKP)寄存器组读写时, 需要对如下寄存器位进行配置: 1.OSCCTL0 寄存器的 PMWREN 位置 1, 允许整个备份域的读写操作; 2.PM_CTL0 寄存器的 BKPREGCLR 位置 1, 寄存器组退出复位; 3.PM_CTL0 寄存器的 BKPWR 位置 1, 允许备份域数据寄存器组和 RTC 模块内的寄存器读写操作。读写寄存器组前调用该函数, 结束读写后必须再次调用该函数, 若缺少或多次调用则会无法恢复原始配置。
3	BKP_Reset_Enable	配置备份域寄存器组软件复位使能
4	BKP_Pin_Effective_Level_Config	设置侵入检测引脚的有效电平
5	BKP_Pin_Enable	设置侵入检测引脚 RTC_TAMP1 使能
6	BKP_RTC_Clock_Config	配置 RTC 时钟源
7	BKP_External_Clock_Bypass_Enable	设置外部时钟旁路控制使能
8	BKP_Data_Config	写备份域备份数据寄存器
9	BKP_Get_Data	读备份域备份数据寄存器
10	BKP_Pin_TAMP_INT_Enable	配置侵入引脚中断使能
11	BKP_Get_Pin_TAMP_INT_Flag	获取侵入中断标志
12	BKP_Clear_Pin_TAMP_INT_Flag	清除侵入检测中断标志

4.4.1 函数 BKP_Reset

表 4-3 函数 BKP_Reset

函数名	BKP_Reset
函数原型	void BKP_Reset (void)

功能描述	复位备份域外设，该函数仅复位备份域寄存器组，而不复位 RTC 模块
输入参数 1	无
返回值	无
被调用函数	无

4.4.2 函数 BKP_Write_And_Read_Enable

表 4-4 函数 BKP_Write_And_Read_Enable

函数名	BKP_Reset_Enable
函数原型	void BKP_Reset_Enable (FunctionalState NewState)
功能描述	备份域(BKP)寄存器组读写时，需要对如下寄存器位进行配置： 1.OSCCTL0 寄存器的 PMWREN 位置 1，允许整个备份域的读写操作； 2.PM_CTL0 寄存器的 BKPREGCLR 位置 1，寄存器组退出复位； 3.PM_CTL0 寄存器的 BKPWR 位置 1，允许备份域数据寄存器组和 RTC 模块内的寄存器读写操作。读写寄存器组前调用该函数，结束读写后必须再次调用该函数，若缺少或多次调用则会无法恢复原始配置
输入参数 1	NewState: 备份域寄存器组读写配置使能，取值范围为： TRUE: 备份域寄存器组读写前的配置 FALSE: 备份域寄存器组读写后的配置恢复
返回值	无
被调用函数	无

4.4.3 函数 BKP_Reset_Enable

表 4-5 函数 BKP_Reset_Enable

函数名	BKP_Reset_Enable
函数原型	void BKP_Reset_Enable (FunctionalState NewState)
功能描述	配置备份域寄存器组软件复位使能
输入参数 1	NewState: 备份域寄存器组软件复位，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

4.4.4 函数 BKP_Pin_Effective_Level_Config

表 4-6 函数 BKP_Pin_Effective_Level_Config

函数名	BKP_Pin_Effective_Level_Config
函数原型	void BKP_Pin_Effective_Level_Config (uint32_t PinSel, uint32_t Effective)
功能描述	设置侵入检测引脚的有效电平
输入参数 1	PinSel: 侵入检测引脚选择，取值为： BKP_PIN_RTC_TAMP1:侵入检测引脚

	RTC_TAMP1BKP_PIN_RTC_TAMP2:侵入检测引脚 RTC_TAMP2 BKP_PIN_RTC_TAMP3:侵入检测引脚 RTC_TAMP3
输入参数 2	Effective: 有效电平, 取值为: BKP_HIGH_LEVEL_EFFECTIVE: 高电平会清除所有的数据备份寄存器 BKP_LOW_LEVEL_EFFECTIVE: 低电平会清除所有的数据备份寄存器
返回值	无
被调用函数	无

4.4.5 函数 BKP_Pin_Enable

表 4-7 函数 BKP_Pin_Enable

函数名	BKP_Pin_Enable
函数原型	void BKP_Pin_Enable (uint32_t PinSel, FunctionalState NewState)
功能描述	设置侵入检测引脚 RTC_TAMP1 使能
输入参数 1	PinSel: 侵入检测引脚选择, 取值为: BKP_PIN_RTC_TAMP1: 侵入检测引脚 RTC_TAMP1 BKP_PIN_RTC_TAMP2: 侵入检测引脚 RTC_TAMP2 BKP_PIN_RTC_TAMP3: 侵入检测引脚 RTC_TAMP3
输入参数 2	NewState: 侵入检测引脚 RTC_TAMP1 使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

4.4.6 函数 BKP_RTC_Clock_Config

表 4-8 函数 BKP_RTC_Clock_Config

函数名	BKP_RTC_Clock_Config
函数原型	void BKP_RTC_Clock_Config (uint32_t Source)
功能描述	配置 RTC 时钟源
输入参数 1	Source: RTC 时钟源选择, 取值为: BKP_RTC_NO_CLK: 无时钟 BKP_RTC_EXTLTF: EXTLF 作为 RTC 时钟 BKP_RTC_INTLTF: INTLTF 作为 RTC 时钟 BKP_RTC_EXTHF_DIV_128: EXTHF 经过 128 分频后作为 RTC 时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

4.4.7 函数 BKP_External_Clock_Bypass_Enable

表 4-9 函数 BKP_External_Clock_Bypass_Enable

函数名	BKP_External_Clock_Bypass_Enable
函数原型	void BKP_External_Clock_Bypass_Enable (uint32_t Source)
功能描述	设置外部时钟旁路控制使能
输入参数 1	Source: 时钟选择, 取值为: BKP_EXTHF: 外部高频时钟 BKP_EXTLF: 外部低频时钟
输入参数 2	NewState: 外部时钟旁路控制使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

4.4.8 函数 BKP_Data_Config

表 4-10 函数 BKP_Data_Config

函数名	BKP_Data_Config
函数原型	void BKP_Data_Config (uint32_t OrderNumber, uint32_t WriteData)
功能描述	写备份域备份数据寄存器
输入参数 1	OrderNumber: 备份数据寄存器编号, 取值范围为 0~31
输入参数 2	WriteData: 写入的数据, 取值范围为 32 位数据
返回值	无
被调用函数	无

4.4.9 函数 BKP_Get_Data

表 4-11 函数 BKP_Get_Data

函数名	BKP_Get_Data
函数原型	uint32_t BKP_Get_Data (uint32_t OrderNumber)
功能描述	读备份域备份数据寄存器
输入参数 1	OrderNumber: 备份数据寄存器编号, 取值范围为 0~31
返回值	备份数据寄存器的值, 取值范围为 32 位数据
被调用函数	无

4.4.10 函数 BKP_Pin_TAMP_INT_Enable

表 4-12 函数 BKP_Get_Data

函数名	BKP_Pin_TAMP_INT_Enable
函数原型	void BKP_Pin_TAMP_INT_Enable (uint32_t OrderNumber)
功能描述	配置侵入引脚中断使能

输入参数 1	PinSel: 侵入检测引脚选择, 取值为: BKP_PIN_RTC_TAMP1: 侵入检测引脚 RTC_TAMP1 BKP_PIN_RTC_TAMP2: 侵入检测引脚 RTC_TAMP2 BKP_PIN_RTC_TAMP3: 侵入检测引脚 RTC_TAMP3
输入参数 2	NewState: 侵入引脚中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

4.4.11 函数 BKP_Get_Pin_TAMP_INT_Flag

表 4-13 函数 BKP_Get_Pin_TAMP_INT_Flag

函数名	BKP_Get_Pin_TAMP_INT_Flag
函数原型	FlagStatus BKP_Get_Pin_TAMP_INT_Flag (uint32_t PinSel)
功能描述	获取侵入中断标志
输入参数 1	PinSel: 侵入检测引脚选择, 取值为: BKP_PIN_RTC_TAMP1: 侵入检测引脚 RTC_TAMP1 BKP_PIN_RTC_TAMP2: 侵入检测引脚 RTC_TAMP2 BKP_PIN_RTC_TAMP3: 侵入检测引脚 RTC_TAMP3
返回值	1:发生侵入事件, 0:无侵入事件发生
被调用函数	无

4.4.12 函数 BKP_Clear_Pin_TAMP_INT_Flag

表 4-14 函数 BKP_Clear_Pin_TAMP_INT_Flag

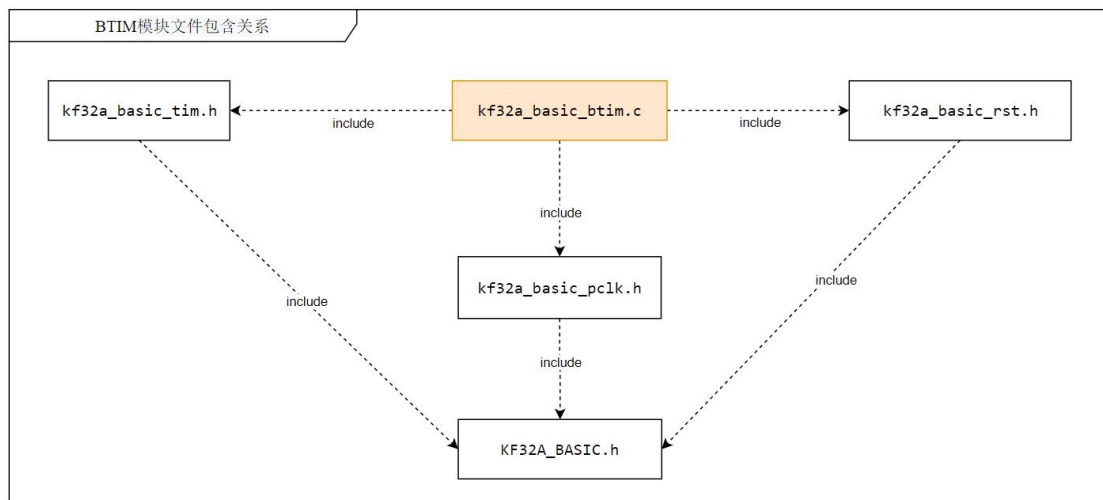
函数名	BKP_Clear_Pin_TAMP_INT_Flag
函数原型	Void BKP_Clear_Pin_TAMP_INT_Flag (uint32_t PinSel)
功能描述	清除侵入检测中断标志
输入参数 1	PinSel: 侵入检测引脚选择, 取值为: BKP_PIN_RTC_TAMP1: 侵入检测引脚 RTC_TAMP1 BKP_PIN_RTC_TAMP2: 侵入检测引脚 RTC_TAMP2 BKP_PIN_RTC_TAMP3: 侵入检测引脚 RTC_TAMP3
返回值	无
被调用函数	无

5 基本定时器(BTIM)

KungFu32 内核提供了两个 16 位的基本定时器（Basic Timer，以下简称 BTIM），基本定时器有定时和计数两种工作模式，支持 3 种计数方式：向上计数、向下计数和向上向下计数方式。根据不同的模式，计数会产生溢出，将 Tx 溢出中断标志 TXIF 位置 1。

5.1 文件引用关系

图 5-1 BTIM 模块文件包含关系



5.2 BTIM 寄存器结构

BTIM 寄存器结构，BTIM_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct BTIM_MemMap {
    volatile uint32_t    CNT;
    volatile uint32_t    CTL1;
    volatile uint32_t    CTL2;
    volatile uint32_t    PRSC;
    volatile uint32_t    PPX;
    volatile uint32_t    DIER;
    volatile const uint32_t    SR;
    volatile uint32_t    SRIC;
} BTIM_SFRmap;
    
```

表 5-1 BTIM 寄存器结构说明

寄存器	描述
CNT	TxCNT 寄存器，偏移:0x00
CTL1	Tx 控制寄存器 1，偏移:0x04

CTL2	Tx 控制寄存器 2, 偏移:0x08
PRSC	Tx 预分频寄存器, 偏移:0x0C
PPX	PPx 周期寄存器, 偏移:0x10
DIER	Tx 中断使能控制寄存器, 偏移:0x14
SR	Tx 中断状态寄存器, 偏移:0x18
SRIC	Tx 中断状态清除寄存器, 偏移:0x1C

BTIM 配置信息结构体, BTIM_SFRmap, 定义于文件 kf32a_basic_tim.h 中, 如下:

```
typedef struct
{
    uint16_t    m_Counter;
    uint16_t    m_Period;
    uint16_t    m_Prescaler;
    uint16_t    m_CounterMode;
    uint16_t    m_Clock;
    uint16_t    m_WorkMode;
    uint16_t    m_MasterMode;
    uint16_t    m_SlaveMode;
    uint16_t    m_EXPulseSync;
    uint16_t    m_MasterSlaveSync;
} BTIM_InitTypeDef;
```

表 2-2 BTIM 配置信息结构体说明

寄存器	描述
m_Counter	定时器计数值, 取值 16 位数据
m_Period	定时器周期值, 取值 16 位数据
m_Prescaler	定时器预分频值, 取值 16 位数据
m_CounterMode	定时器计数模式, 取值为宏“BTIM 定时器计数模式”中的一个
m_Clock	定时器工作时钟, 取值为宏“BTIM 定时器工作时钟”中的一个
m_WorkMode	定时/计数模式选择取值为宏“BTIM 定时/计数模式选择”中的一个
m_MasterMode	主模式选择, 取值为宏“BTIM 主模式选择”中的一个
m_SlaveMode	从模式选择, 取值为宏“BTIM 从模式选择”中的一个
m_EXPulseSync	Tx 计数模式外部触发脉冲输入同步控制, 取值为宏“BTIM 计数模式外部触发脉冲输入同步控制”中的一个
m_MasterSlaveSync	从模式选择, 取值为宏“BTIM 从模式选择”中的一个

5.3 BTIM 宏定义

BTIM 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, BTIM 其他相关宏定义详见文件 kf32a_basic_tim.h 及函数参数描述。

5.4 BTIM 库函数

表 5-2 BTIM 固件库函数列表

序号	函数名	描述
1	TIM_Reset	定时器外设复位，使能外设时钟
2	BTIM_Configuration	基本定时器(BTIM)配置
3	BTIM_Struct_Init	初始化基本定时器配置信息结构体
4	BTIM_Cmd	定时器启动控制使能
5	BTIM_Set_Counter	更新定时器计数值
6	BTIM_Set_Period	更新定时器周期值
7	BTIM_Set_Prescaler	更新定时器预分频值
8	BTIM_Counter_Mode_Config	更新定时器计数模式
9	BTIM_Clock_Config	更新定时器工作时钟
10	BTIM_External_Pulse_Sync_Config	更新定时器计数模式外部触发脉冲输入同步控制位
11	BTIM_Work_Mode_Config	更新定时/计数模式选择
12	BTIM_Generate_Trigger_Config	更新产生触发事件配置位
13	BTIM_Single_Pulse_Enable	配置单脉冲输出模式
14	BTIM_Single_Pulse_Shut_Enable	配置单脉冲输出模式
15	BTIM_Update_Immediately_Config	配置立即更新控制位
16	BTIM_Master_Slave_Sync_Config	配置主从模式同步位
17	BTIM_Trigger_Select_Config	配置触发选择位
18	BTIM_Slave_Mode_Config	配置从模式选择位
19	BTIM_Master_Mode_Config	配置主模式选择位
20	BTIM_Update_Rising_Edge_Config	配置上升沿更新事件控制位
21	BTIM_Update_Enable	配置更新使能
22	BTIM_Get_Direction	读 TX 计数方向
23	BTIM_Get_Counter	读定时器计数值
24	BTIM_Get_Period	读定时器周期值
25	BTIM_Get_Prescaler	读定时器预分频值
26	BTIM_Trigger_DMA_Enable	配置触发事件的 DMA 请求使能
27	BTIM_Update_DMA_Enable	配置更新事件的 DMA 请求使能
28	BTIM_Overflow_INT_Enable	配置 Tx 计数溢出中断使能
29	BTIM_Trigger_INT_Enable	配置 Tx 触发事件中断使能
30	BTIM_Update_INT_Enable	配置 Tx 更新事件中断使能
31	BTIM_Get_Trigger_DMA_INT_Status	配置触发事件的 DMA 请求使能
32	BTIM_Get_Update_DMA_INT_Status	配置更新事件的 DMA 请求使能
33	BTIM_Get_Overflow_INT_Status	配置 Tx 计数溢出中断使能
34	BTIM_Get_Trigger_INT_Status	配置 Tx 触发事件中断使能
35	BTIM_Get_Update_INT_Status	配置 Tx 更新事件中断使能

36	BTIM_Get_Trigger_DMA_INT_Flag	获取触发事件触发 DMA 中断标志
37	BTIM_Get_Updata_DMA_INT_Flag	获取更新事件触发 DMA 中断标志
38	BTIM_Get_Overflow_INT_Flag	获取 Tx 计数溢出中断标志
39	BTIM_Get_Trigger_INT_Flag	获取 Tx 触发事件中断标志
40	BTIM_Get_Updata_INT_Flag	获取 Tx 更新事件中断标志
41	BTIM_Clear_Overflow_INT_Flag	清除 Tx 计数溢出中断标志
42	BTIM_Clear_Trigger_INT_Flag	清除 Tx 触发事件中断标志
43	BTIM_Clear_Updata_INT_Flag	清除 Tx 更新事件中断标志

5.4.1 函数 TIM_Reset

表 5-3 函数 TIM_Reset

函数名	TIM_Reset
函数原型	void TIM_Reset (void* TIMx)
功能描述	定时器外设复位，使能外设时钟
输入参数 1	TIMx: 定时器内存结构指针，取值为 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T5_SFR/T6_SFR/T7_SFR/T8_SFR/T9_SFR/T10_SFR/T14_SFR/T15_SFR/T18_SFR/T19_SFR/T20_SFR/T21_SFR/T22_SFR/T23_SFR
返回值	无
被调用函数	RST_CTL1_Peripheral_Reset_Enable(RST_CTL1_T0RST, TRUE); RST_CTL1_Peripheral_Reset_Enable(RST_CTL1_T0RST, FALSE); PCLK_CTL1_Peripheral_Clock_Enable(PCLK_CTL1_T0CLKEN, TRUE);

5.4.2 函数 BTIM_Configuration

表 5-4 函数 BTIM_Configuration

函数名	BTIM_Configuration
函数原型	void BTIM_Configuration (BTIM_SFRmap* BTIMx,BTIM_InitTypeDef* btimInitStruct)
功能描述	基本定时器(BTIM)配置
输入参数 1	BTIMx: 指向定时器内存结构的指针，取值 T14_SFR、T15_SFR
输入参数 2	btimInitStruct: 基本定时器配置信息结构体指针
返回值	无
被调用函数	无

5.4.3 函数 BTIM_Struct_Init

表 5-5 函数 BTIM_Struct_Init

函数名	BTIM_Struct_Init
函数原型	void BTIM_Struct_Init (BTIM_InitTypeDef* btimInitStruct)

功能描述	初始化基本定时器配置信息结构体
输入参数 1	btimInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

5.4.4 函数 BTIM_Cmd

表 5-6 函数 BTIM_Cmd

函数名	BTIM_Cmd
函数原型	void BTIM_Cmd (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	定时器启动控制使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 定时器使能控制, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.5 函数 BTIM_Set_Counter

表 5-7 函数 BTIM_Set_Counter

函数名	BTIM_Set_Counter
函数原型	void BTIM_Set_Counter (BTIM_SFRmap* BTIMx, uint16_t Counter)
功能描述	更新定时器计数值
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	Counter: 新的计数值, 取值 16 位数据
返回值	无
被调用函数	无

5.4.6 函数 BTIM_Set_Period

表 5-8 函数 BTIM_Set_Period

函数名	BTIM_Set_Period
函数原型	void BTIM_Set_Period (BTIM_SFRmap* BTIMx, uint16_t Period)
功能描述	更新定时器周期值
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	Period: 新的周期值, 取值 16 位数据
返回值	无
被调用函数	无

5.4.7 函数 BTIM_Set_Prescaler

表 5-9 函数 BTIM_Set_Prescaler

函数名	BTIM_Set_Prescaler
函数原型	void BTIM_Set_Prescaler (BTIM_SFRmap* BTIMx, uint16_t Prescaler)
功能描述	更新定时器预分频值
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	Prescaler: 新的预分频, 取值 16 位数据
返回值	无
被调用函数	无

5.4.8 函数 BTIM_Counter_Mode_Config

表 5-10 函数 BTIM_Counter_Mode_Config

函数名	BTIM_Counter_Mode_Config
函数原型	void BTIM_Counter_Mode_Config (BTIM_SFRmap* BTIMx, uint32_t CounterMode)
功能描述	更新定时器计数模式
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	CounterMode: 新的计数模式, 取值范围为: BTIM_COUNT_DOWN_UF: 向下计数, 下溢产生中断标志 BTIM_COUNT_UP_OF: 向上计数, 上溢产生中断标志 BTIM_COUNT_UP_DOWN_OF: 向上-向下计数, 上溢产生中断标志 BTIM_COUNT_UP_DOWN_UF: 向上-向下计数, 下溢产生中断标志 BTIM_COUNT_UP_DOWN_OUF: 向上-向下计数, 上溢和下溢产生中断标志
返回值	无
被调用函数	无

5.4.9 函数 BTIM_Clock_Config

表 5-11 函数 BTIM_Clock_Config

函数名	BTIM_Clock_Config
函数原型	void BTIM_Clock_Config (BTIM_SFRmap* BTIMx, uint32_t NewClock)
功能描述	更新定时器工作时钟
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewClock: 新的工作时钟, 取值范围为: BTIM_SCLK: 选用 SCLK 时钟 BTIM_HFCLK: 选用 HFCLK 时钟 BTIM_LFCLK: 选用 LFCLK 时钟

返回值	无
被调用函数	无

5.4.10 函数 BTIM_External_Pulse_Sync_Config

表 5-12 函数 BTIM_External_Pulse_Sync_Config

函数名	BTIM_External_Pulse_Sync_Config
函数原型	void BTIM_External_Pulse_Sync_Config (BTIM_SFRmap* BTIMx, uint32_t PulseSync)
功能描述	更新定时器计数模式外部触发脉冲输入同步控制位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewClock: 新的定时器计数模式外部触发脉冲输入同步控制位使能状态, 取值范围为: BTIM_EX_SYNC_MODE: 与外部触发脉冲输入同步 BTIM_NO_SYNC_MODE: 不与外部触发脉冲输入同步
返回值	无
被调用函数	无

5.4.11 函数 BTIM_Work_Mode_Config

表 5-13 函数 BTIM_Work_Mode_Config

函数名	BTIM_Work_Mode_Config
函数原型	void BTIM_Work_Mode_Config (BTIM_SFRmap* BTIMx, uint32_t NewState)
功能描述	更新定时/计数模式选择
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 新的定时/计数模式, 取值范围为: BTIM_TIMER_MODE: 定时模式 BTIM_COUNTER_MODE: 计数模式
返回值	无
被调用函数	无

5.4.12 函数 BTIM_Generate_Trigger_Config

表 5-14 函数 BTIM_Generate_Trigger_Config

函数名	BTIM_Generate_Trigger_Config
函数原型	void BTIM_Generate_Trigger_Config (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	更新产生触发事件配置位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR

输入参数 2	NewState: 定时器使能控制状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.13 函数 BTIM_Single_Pulse_Enable

表 5-15 函数 BTIM_Single_Pulse_Enable

函数名	BTIM_Single_Pulse_Enable
函数原型	void BTIM_Single_Pulse_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置单脉冲输出模式
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 单脉冲输出模式使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.14 函数 BTIM_Single_Pulse_Shut_Enable

表 5-16 函数 BTIM_Single_Pulse_Shut_Enable

函数名	BTIM_Single_Pulse_Shut_Enable
函数原型	void BTIM_Single_Pulse_Shut_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置单脉冲输出模式
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 单脉冲输出模式选择, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.15 函数 BTIM_Updata_Immediately_Config

表 5-17 函数 BTIM_Updata_Immediately_Config

函数名	BTIM_Updata_Immediately_Config
函数原型	void BTIM_Updata_Immediately_Config (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置立即更新控制位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 立即更新使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.16 函数 BTIM_Master_Slave_Snyc_Config

表 5-18 函数 BTIM_Master_Slave_Snyc_Config

函数名	BTIM_Master_Slave_Snyc_Config
函数原型	void BTIM_Master_Slave_Snyc_Config (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置主从模式同步位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 主从模式同步位状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.17 函数 BTIM_Trigger_Select_Config

表 5-19 函数 BTIM_Trigger_Select_Config

函数名	BTIM_Trigger_Select_Config
函数原型	void BTIM_Trigger_Select_Config (BTIM_SFRmap* BTIMx, uint32_t TriggerSelect)
功能描述	配置触发选择位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	TriggerSelect: 触发选择, 取值范围为: BTIM_TRIGGER_T1 BTIM_TRIGGER_T2 BTIM_TRIGGER_T3 BTIM_TRIGGER_T4 BTIM_TRIGGER_T5 BTIM_TRIGGER_T9 BTIM_TRIGGER_T14 BTIM_TRIGGER_T15 BTIM_TRIGGER_T18 BTIM_TRIGGER_T19 BTIM_TRIGGER_T20 BTIM_TRIGGER_T21 BTIM_TRIGGER_TXCK
返回值	无
被调用函数	无

5.4.18 函数 BTIM_Slave_Mode_Config

表 5-20 函数 BTIM_Slave_Mode_Config

函数名	BTIM_Slave_Mode_Config
-----	------------------------

函数原型	void BTIM_Slave_Mode_Config (BTIM_SFRmap* BTIMx, uint32_t SlaveMode)
功能描述	配置从模式选择位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	SlaveSelect: 从模式选择, 取值范围为: BTIM_SLAVE_FORBIDDEN_MODE: 从模式禁止 BTIM_SLAVE_TRIGGER_MODE: 触发模式 BTIM_SLAVE_GATED_MODE: 门控模式 BTIM_SLAVE_RESET_MODE: 复位模式 BTIM_SLAVE_COUNTER_MODE: 计数模式 2
返回值	无
被调用函数	无

5.4.19 函数 BTIM_Master_Mode_Config

表 5-21 函数 BTIM_Master_Mode_Config

函数名	BTIM_Master_Mode_Config
函数原型	void BTIM_Master_Mode_Config (BTIM_SFRmap* BTIMx, uint32_t MasterMode)
功能描述	配置主模式选择位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	MasterMode: 主模式选择, 取值范围为: BTIM_MASTER_UR_SIGNAL: UR 位作为触发 BTIM_MASTER_EN_SIGNAL: TXEN 作为触发 BTIM_MASTER_IF_SIGNAL: TXIF 作为触发
返回值	无
被调用函数	无

5.4.20 函数 BTIM_Updata_Rising_Edge_Config

表 5-22 函数 BTIM_Updata_Rising_Edge_Config

函数名	BTIM_Updata_Rising_Edge_Config
函数原型	void BTIM_Updata_Rising_Edge_Config (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置上升沿更新事件控制位
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 更新事件控制状态, 取值范围为: TRUE: 上升沿立即更新 FALSE: 每周期更新
返回值	无

被调用函数	无
-------	---

5.4.21 函数 BTIM_Update_Enable

表 5-23 函数 BTIM_Update_Enable

函数名	BTIM_Update_Enable
函数原型	void BTIM_Update_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置更新使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 更新使能状态, 取值范围为: TRUE: 允许更新 FALSE: 禁止更新
返回值	无
被调用函数	无

5.4.22 函数 BTIM_Get_Direction

表 5-24 函数 BTIM_Get_Direction

函数名	BTIM_Get_Direction
函数原型	DIRStatus BTIM_Get_Direction (BTIM_SFRmap* BTIMx)
功能描述	读 TX 计数方向
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	TX 计数方向, 0: 向下, 1: 向上
被调用函数	无

5.4.23 函数 BTIM_Get_Counter

表 5-25 函数 BTIM_Get_Counter

函数名	BTIM_Get_Counter
函数原型	uint16_t BTIM_Get_Counter (BTIM_SFRmap* BTIMx)
功能描述	读定时器计数值
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	Tx 计数值, 16 位
被调用函数	无

5.4.24 函数 BTIM_Get_Period

表 5-26 函数 BTIM_Get_Period

函数名	BTIM_Get_Period
函数原型	uint16_t BTIM_Get_Period (BTIM_SFRmap* BTIMx)
功能描述	读定时器周期值
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	Tx 周期值, 16 位
被调用函数	无

5.4.25 函数 BTIM_Get_Prescaler

表 5-27 函数 BTIM_Get_Prescaler

函数名	BTIM_Get_Prescaler
函数原型	uint16_t BTIM_Get_Prescaler (BTIM_SFRmap* BTIMx)
功能描述	读定时器预分频值
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	Tx 预分频值, 16 位
被调用函数	无

5.4.26 函数 BTIM_Trigger_DMA_Enable

表 5-28 函数 BTIM_Trigger_DMA_Enable

函数名	BTIM_Trigger_DMA_Enable
函数原型	void BTIM_Trigger_DMA_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置触发事件的 DMA 请求使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: 触发事件的 DMA 请求, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.27 函数 BTIM_Updata_DMA_Enable

表 5-29 函数 BTIM_Updata_DMA_Enable

函数名	BTIM_Updata_DMA_Enable
函数原型	void BTIM_Updata_DMA_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置更新事件的 DMA 请求使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR

输入参数 2	NewState: 更新事件的 DMA 请求, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.28 函数 BTIM_Overflow_INT_Enable

表 5-30 函数 BTIM_Overflow_INT_Enable

函数名	BTIM_Overflow_INT_Enable
函数原型	void BTIM_Update_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置 Tx 计数溢出中断使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: Tx 计数溢出中断, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.29 函数 BTIM_Trigger_INT_Enable

表 5-31 函数 BTIM_Trigger_INT_Enable

函数名	BTIM_Trigger_INT_Enable
函数原型	void BTIM_Trigger_INT_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置 Tx 触发事件中断使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: Tx 触发事件中断, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.30 函数 BTIM_Update_INT_Enable

表 5-32 函数 BTIM_Update_INT_Enable

函数名	BTIM_Update_INT_Enable
函数原型	void BTIM_Update_INT_Enable (BTIM_SFRmap* BTIMx, FunctionalState NewState)
功能描述	配置 Tx 更新事件中断使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
输入参数 2	NewState: Tx 更新事件中断, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

5.4.31 函数 BTIM_Get_Trigger_DMA_INT_Status

表 5-33 函数 BTIM_Update_Enable

函数名	BTIM_Update_Enable
函数原型	INTStatus BTIM_Get_Trigger_DMA_INT_Status (BTIM_SFRmap* BTIMx)
功能描述	配置触发事件的 DMA 请求使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断使能状态, 0: 未使能, 1: 使能
被调用函数	无

5.4.32 函数 BTIM_Get_Update_DMA_INT_Status

表 5-34 函数 BTIM_Get_Update_DMA_INT_Status

函数名	BTIM_Get_Update_DMA_INT_Status
函数原型	INTStatus BTIM_Get_Update_DMA_INT_Status (BTIM_SFRmap* BTIMx)
功能描述	配置更新事件的 DMA 请求使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断使能状态, 0: 未使能, 1: 使能
被调用函数	无

5.4.33 函数 BTIM_Get_Overflow_INT_Status

表 5-35 函数 BTIM_Get_Overflow_INT_Status

函数名	BTIM_Get_Overflow_INT_Status
函数原型	INTStatus BTIM_Get_Overflow_INT_Status (BTIM_SFRmap* BTIMx)
功能描述	配置 Tx 计数溢出中断使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断使能状态, 0: 未使能, 1: 使能
被调用函数	无

5.4.34 函数 BTIM_Get_Trigger_INT_Status

表 5-36 函数 BTIM_Get_Trigger_INT_Status

函数名	BTIM_Get_Trigger_INT_Status
函数原型	INTStatus BTIM_Get_Trigger_INT_Status (BTIM_SFRmap* BTIMx)
功能描述	配置 Tx 触发事件中断使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断使能状态, 0: 未使能, 1: 使能
被调用函数	无

5.4.35 函数 BTIM_Get_Updata_INT_Status

表 5-37 函数 BTIM_Get_Updata_INT_Status

函数名	BTIM_Get_Updata_INT_Status
函数原型	INTStatus BTIM_Get_Updata_INT_Status (BTIM_SFRmap* BTIMx)
功能描述	配置 Tx 更新事件中断使能
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断使能状态, 0: 未使能, 1: 使能
被调用函数	无

5.4.36 函数 BTIM_Get_Trigger_DMA_INT_Flag

表 5-38 函数 BTIM_Get_Trigger_DMA_INT_Flag

函数名	BTIM_Get_Trigger_DMA_INT_Flag
函数原型	FlagStatus BTIM_Get_Trigger_DMA_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	获取触发事件触发 DMA 中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

5.4.37 函数 BTIM_Get_Updata_DMA_INT_Flag

表 5-39 函数 BTIM_Get_Updata_DMA_INT_Flag

函数名	BTIM_Get_Updata_DMA_INT_Flag
函数原型	FlagStatus BTIM_Get_Updata_DMA_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	获取更新事件触发 DMA 中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

5.4.38 函数 BTIM_Get_Overflow_INT_Flag

表 5-40 函数 BTIM_Get_Overflow_INT_Flag

函数名	BTIM_Get_Overflow_INT_Flag
函数原型	FlagStatus BTIM_Get_Overflow_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	获取 Tx 计数溢出中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

5.4.39 函数 BTIM_Get_Update_INT_Flag

表 5-41 函数 BTIM_Get_Update_INT_Flag

函数名	BTIM_Get_Update_INT_Flag
函数原型	FlagStatus BTIM_Get_Update_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	获取 Tx 更新事件中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

5.4.40 函数 BTIM_Clear_Overflow_INT_Flag

表 5-42 函数 BTIM_Clear_Overflow_INT_Flag

函数名	BTIM_Clear_Overflow_INT_Flag
函数原型	void BTIM_Clear_Overflow_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	清除 Tx 计数溢出中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	无
被调用函数	无

5.4.41 函数 BTIM_Clear_Trigger_INT_Flag

表 5-43 函数 BTIM_Clear_Trigger_INT_Flag

函数名	BTIM_Clear_Trigger_INT_Flag
函数原型	void BTIM_Clear_Trigger_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	清除 Tx 触发事件中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	无
被调用函数	无

5.4.42 函数 BTIM_Clear_Update_INT_Flag

表 5-44 函数 BTIM_Clear_Update_INT_Flag

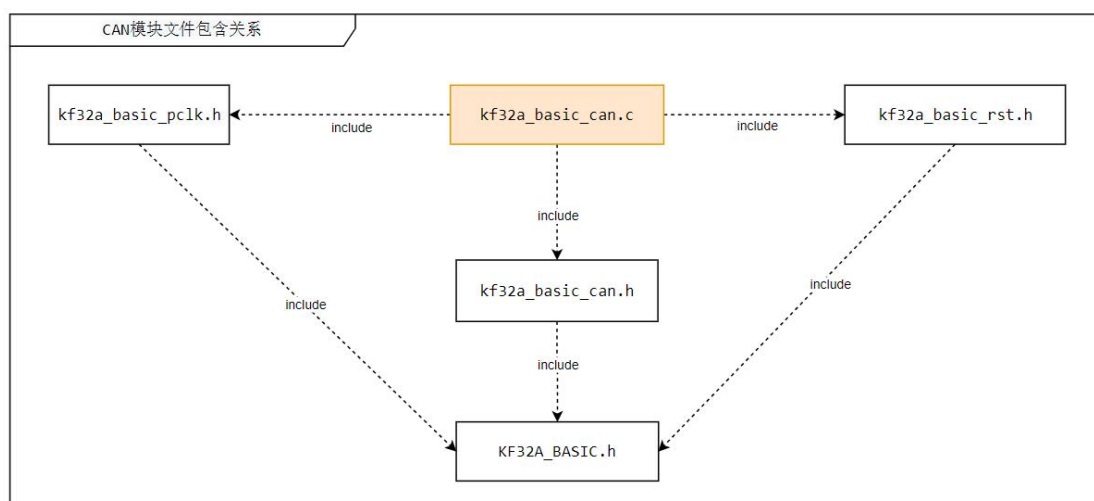
函数名	BTIM_Clear_Update_INT_Flag
函数原型	void BTIM_Clear_Update_INT_Flag (BTIM_SFRmap* BTIMx)
功能描述	清除 Tx 更新事件中断标志
输入参数 1	BTIMx: 指向定时器内存结构的指针, 取值 T14_SFR、T15_SFR
返回值	无
被调用函数	无

6 控制器局域网（CAN）

控制器局域网（CAN）是一种多节点双绞线通信网络，具有较强的抗电磁干扰特性。CAN 总线长度小于 40 米时，最高可达 1Mbps 位速率。

6.1 文件引用关系

图 6-1 IWDWT 模块文件包含关系



6.2 CAN 寄存器结构体

CAN 配置信息结构体，CAN_InitTypeDef，定义于文件 kf32a_basic_can.h 中，如下：

```

typedef struct
{
    FunctionalState    m_Enable;
    uint32_t          m_Mode;
    uint32_t          m_WorkSource;
    uint8_t           m_BaudRate;
    uint8_t           m_SyncJumpWidth;
    uint8_t           m_TimeSeg1;
    uint8_t           m_TimeSeg2;
    uint32_t          m_BusSample;
    uint32_t          m_Acceptance;
    uint32_t          m_AcceptanceMask;
}CAN_InitTypeDef;
    
```

表 6-1 CAN_InitTypeDef 寄存器结构说明

寄存器	描述
-----	----

m_Enable	CAN 使能选择, 取值为 TRUE 或 FALSE
m_Mode	CAN 工作模式, 取值宏 “CAN 工作模式” 中的一个
m_WorkSource	CAN 工作时钟, 取值宏 “CAN 工作时钟” 中的一个
m_BaudRate	CAN 波特率预设值, 取值为 0~0x3F
m_SyncJumpWidth	CAN 同步跳转宽度, 取值为 0~0x3
m_TimeSeg1	CAN 时间段 1, 取值为 0~0xF
m_TimeSeg2	CAN 时间段 2, 取值为 0~0x7
m_BusSample	CAN 总线采样次数, 取值宏 “CAN 总线采样次数” 中的一个
m_Acceptance	CAN 验收代码, 取值为 32 位数据
m_AcceptanceMask	CAN 验收掩码

CAN 总线错误信息结构体, CAN_ErrorTypeDef, 定义于文件 kf32a_basic_can.h 中, 如下:

```
typedef struct
{
    uint8_t    m_ErrorCode;
    uint8_t    m_ErrorDirection;
    uint8_t    m_ErrorSegment;
    uint8_t    m_ArbitrationLost;
}CAN_ErrorTypeDef;
```

表 6-2 CAN_ErrorTypeDef 寄存器结构说明

寄存器	描述
m_ErrorCode	错误代码, 取值宏 “CAN 错误代码” 中的一个
ErrorDirection	传输方向, 取值宏 “CAN 错误传输方向” 中的一个
m_ErrorSegment	错误发生的段, 取值宏 “CAN 错误发生的段” 中的一个
m_ArbitrationLost	仲裁丢失位置, 取值为宏 “CAN 仲裁丢失位置” 中的一个

CAN 报文信息结构体, CAN_MessageTypeDef, 定义于文件 kf32a_basic_can.h 中, 如下:

```
typedef struct
{
    uint32_t    m_FrameFormat;
    uint32_t    m_RemoteTransmit;
    uint32_t    m_DataLength;
    uint32_t    m_StandardID;
    uint32_t    m_ExtendedID;
    uint8_t     m_Data[16];
}CAN_MessageTypeDef;
```

表 6-3 CAN_MessageTypeDef 寄存器结构说明

寄存器	描述
m_FrameFormat	帧格式选择, 取值宏 “CAN 帧格式” 中的一个
m_RemoteTransmit	远程发送请求, 取值宏 “CAN 远程发送请求” 中的一个

m_DataLength	数据长度，取值为 4 位数据
m_StandardID	标准帧 ID，取值为 0~0x7FF
m_ExtendedID	扩展帧 ID，取值为 0~0x1FFFFFFF
m_Data[16]	数据区，取值为 16 字节数据

6.3 CAN 宏定义

CAN 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，CAN 其他相关宏定义详见文件 kf32a_basic_can.h 及函数参数描述。

6.4 CAN 库函数

表 6-4 CAN 固件库函数列表

函数名	描述
CAN_Reset	复位 CAN 外设，使能外设时钟
CAN_Configuration	控制器局域网总线(CAN)初始化，该函数会在配置中置一复位模式 RSMOD，并在函数结束时清零 RSMOD
CAN_Struct_Init	初始化 CAN 配置信息结构体
CAN_Get_Receive_Message_Counter	获取 CAN RX 信息计数器
CAN_Get_Transmit_Status	获取 CAN 传输事件状态类型
CAN_Cmd	配置 CAN 使能
CAN_Clock_Source_Config	配置 CAN 工作时钟选择
CAN_Sleep_Mode_Enable	配置 CAN 睡眠模式使能
CAN_Reset_Mode_Enable	配置 CAN 复位模式使能
CAN_Work_Mode_Config	配置 CAN 工作模式
CAN_Bus_Sample_Times_Config	配置 CAN 总线采样次数
CAN_Time_Segment_Config	配置 CAN 两个时间段
CAN_Sync_Jump_Width_Config	配置 CAN 同步跳转宽度
CAN_Baud_Rate_Preset_Config	配置 CAN 波特率
CAN_Get_Error_Code	获取 CAN 总线错误信息
CAN_Get_Error_Warning_Limit	获取 CAN 错误报警限制
CAN_Get_Error_Counter	获取 CAN 发送/接收错误计数器
CAN_Error_Warning_Limit_Config	配置 CAN 错误报警限制
CAN_Error_Counter_Config	配置 CAN 发送/接收错误计数器
CAN_Acceptance_Config	配置 CAN 验收代码
CAN_Get_Acceptance	获取 CAN 验收代码
CAN_Acceptance_Mask_Config	配置 CAN 验收屏蔽
CAN_Get_Acceptance_Mask	获取 CAN 验收屏蔽
CAN_Transmit_Message_Configuration	CAN 模块(CAN)发送缓冲配置

CAN_Receive_Message_Configuration	获取 CAN 模块(CAN)接收缓冲的一个帧信息
CAN_Message_Struct_Init	初始化 CAN 报文信息结构体
CAN_Clear_Buffer_Overflow_Flag	清除 CAN 清除缓冲器满标志
CAN_Release_Receive_Buffer	释放 CAN 接收缓冲器
CAN_Transmit_Single	单次发送,当发生错误或者仲裁丢失时不会进行重发（单次发送）
CAN_Transmit_Repeat	发生错误、仲裁丢失或发送结束时会进行重发（连续发送）
CAN_Frame_Format_Config	配置 CAN 发送缓冲器帧格式
CAN_Remote_Request_Config	配置 CAN 发送缓冲器远程发送请求
CAN_Data_Length_Config	配置 CAN 发送缓冲器数据长度,大于 8 的数据长度代码是不可用的
CAN_Identification_Code_Config	配置 CAN 发送缓冲器识别码
CAN_Get_INT_Flag	获取 CAN 中断标志
CAN_Clear_INT_Flag	清零 CAN 中断标志
CAN_Set_INT_Enable	配置 CAN 中断使能

6.4.1 函数 CAN_Reset

表 6-5 函数 CAN_Reset

函数名	CAN_Reset
函数原型	void CAN_Reset(CAN_SFRmap* CANx)
功能描述	复位 CAN 外设,使能外设时钟
输入参数 1	CANx: 指向 CAN 内存结构的指针,取值为 CAN0_SFR~CAN5_SFR
返回值	无
被调用函数	无

6.4.2 函数 CAN_Configuration

表 6-6 函数 CAN_Configuration

函数名	CAN_Configuration
函数原型	void CAN_Configuration(CAN_SFRmap* CANx, CAN_InitTypeDef* canInitStruct)
功能描述	控制器局域网总线(CAN)初始化,该函数会在配置中置一复位模式 RSMOD,并在函数结束时清零 RSMOD
输入参数 1	CANx: 指向 CAN 内存结构的指针,取值为 CAN0_SFR~CAN5_SFR
输入参数 2	canInitStruct: CAN 配置信息结构体
返回值	无
被调用函数	无

6.4.3 函数 CAN_Struct_Init

表 6-7 函数 CAN_Struct_Init

函数名	CAN_Struct_Init
函数原型	void CAN_Struct_Init (CAN_InitTypeDef* canInitStruct)
功能描述	初始化 CAN 配置信息结构体
输入参数 1	canInitStruct: CAN 配置信息结构体
返回值	无
被调用函数	无

6.4.4 函数 CAN_Get_Receive_Message_Counter

表 6-8 函数 CAN_Get_Receive_Message_Counter

函数名	CAN_Get_Receive_Message_Counter
函数原型	uint32_t CAN_Get_Receive_Message_Counter (CAN_SFRmap* CANx)
功能描述	获取 CAN RX 信息计数器
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
返回值	RX 信息计数器, 有效值为 5 位
被调用函数	无

6.4.5 函数 CAN_Get_Transmit_Status

表 6-9 函数 CAN_Get_Transmit_Status

函数名	CAN_Get_Transmit_Status
函数原型	FlagStatus CAN_Get_Transmit_Status (CAN_SFRmap* CANx, uint32_t Type)
功能描述	获取 CAN 传输事件状态类型
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
返回值	1: 总线关闭/出错/正在发送/正在接收/发送完毕/CPU 可以访问发送缓冲器/ RAM 中未读数据满/RAM 中有未读信息 0: 总线开启/无错/发送空闲/接收空闲/发送未完成/CPU 不能访问发送缓冲 器 RAM 中未读数据未读/RAM 中无未读信息
被调用函数	无

6.4.6 函数 CAN_Cmd

表 6-10 函数 CAN_Cmd

函数名	CAN_Cmd
函数原型	void CAN_Cmd (CAN_SFRmap* CANx, FunctionalState NewState)
功能描述	配置 CAN 使能
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR

输入参数 2	NewState: CAN 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

6.4.7 函数 CAN_Clock_Source_Config

表 6-11 函数 CAN_Clock_Source_Config

函数名	CAN_Clock_Source_Config
函数原型	void CAN_Clock_Source_Config (CAN_SFRmap* CANx, uint32_t ClockSource)
功能描述	配置 CAN 工作时钟选择
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	ClockSource: CAN 工作时钟, 取值为: CAN_SOURCE_SCLK_DIV_2: 选择 SCLK 主时钟/2 作为 CAN 工作时钟 CAN_SOURCE_HFCLK_DIV_2: 选择 HFCLK 时钟/2 作为 CAN 工作时钟 CAN_SOURCE_LFCLK_DIV_2: 选择 LFCLK 时钟/2 作为 CAN 工作时钟
返回值	无
被调用函数	无

6.4.8 函数 CAN_Sleep_Mode_Enable

表 6-12 函数 CAN_Sleep_Mode_Enable

函数名	CAN_Sleep_Mode_Enable
函数原型	void CAN_Sleep_Mode_Enable (CAN_SFRmap* CANx, FunctionalState NewState)
功能描述	配置 CAN 睡眠模式使能
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	NewState: CAN 睡眠模式使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

6.4.9 函数 CAN_Reset_Mode_Enable

表 6-13 函数 CAN_Reset_Mode_Enable

函数名	CAN_Reset_Mode_Enable
函数原型	void CAN_Reset_Mode_Enable (CAN_SFRmap* CANx, FunctionalState NewState)
功能描述	配置 CAN 复位模式使能
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	NewState: CAN 睡眠模式使能状态, 取值为 TRUE 或 FALSE

返回值	无
被调用函数	无

6.4.10 函数 CAN_Work_Mode_Config

表 6-14 函数 CAN_Work_Mode_Config

函数名	CAN_Work_Mode_Config
函数原型	void CAN_Work_Mode_Config (CAN_SFRmap* CANx, uint32_t ModeType)
功能描述	配置 CAN 复位模式使能
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	ModeType: CAN 四种模式, 取值为: CAN_MODE_NORMAL: 禁止安静和回环模式 CAN_MODE_SILENT: 使能安静模式 CAN_MODE_LOOPBACK: 使能回环模式 CAN_MODE_SILENT_LOOPBACK: 使能安静和回环模式
返回值	无
被调用函数	无

6.4.11 函数 CAN_Bus_Sample_Times_Config

表 6-15 函数 CAN_Bus_Sample_Times_Config

函数名	CAN_Bus_Sample_Times_Config
函数原型	void CAN_Bus_Sample_Times_Config (CAN_SFRmap* CANx, uint32_t Times)
功能描述	配置 CAN 总线采样次数
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Times: 总线采样次数, 取值为: CAN_BUS_SAMPLE_1_TIME: 总线采样 1 次 CAN_BUS_SAMPLE_3_TIMES: 总线采样 3 次
返回值	无
被调用函数	无

6.4.12 函数 CAN_Time_Segment_Config

表 6-16 函数 CAN_Time_Segment_Config

函数名	CAN_Time_Segment_Config
函数原型	void CAN_Time_Segment_Config (CAN_SFRmap* CANx, uint32_t TimeSeg1, uint32_t TimeSeg2)
功能描述	配置 CAN 两个时间段
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR

输入参数 2	TimeSeg1: CAN 时间段 1, 取值为 4 位有效数值
输入参数 3	TimeSeg2: CAN 时间段 2, 取值为 3 位有效数值
返回值	无
被调用函数	无

6.4.13 函数 CAN_Sync_Jump_Width_Config

表 6-17 函数 CAN_Sync_Jump_Width_Config

函数名	CAN_Sync_Jump_Width_Config
函数原型	void CAN_Sync_Jump_Width_Config (CAN_SFRmap* CANx, uint32_t JumpWidth)
功能描述	配置 CAN 同步跳转宽度
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	JumpWidth: 同步跳转宽度, 取值为 2 位有效数值
返回值	无
被调用函数	无

6.4.14 函数 CAN_Baud_Rate_Preset_Config

表 6-18 函数 CAN_Baud_Rate_Preset_Config

函数名	CAN_Baud_Rate_Preset_Config
函数原型	void CAN_Baud_Rate_Preset_Config (CAN_SFRmap* CANx, uint32_t BaudRate)
功能描述	配置 CAN 波特率
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	BaudRate: 波特率预设值, 取值为 6 位有效数值
返回值	无
被调用函数	无

6.4.15 函数 CAN_Get_Error_Code

表 6-19 函数 CAN_Get_Error_Code

函数名	CAN_Get_Error_Code
函数原型	void CAN_Get_Error_Code (CAN_SFRmap* CANx, CAN_ErrorTypeDef* canErrorStruct)
功能描述	获取 CAN 总线错误信息
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	canErrorStruct: CAN 总线错误信息结构体, 记录错误信息
返回值	无
被调用函数	无

6.4.16 函数 CAN_Get_Error_Warning_Limit

表 6-20 函数 CAN_Get_Error_Warning_Limit

函数名	CAN_Get_Error_Warning_Limit
函数原型	uint8_t CAN_Get_Error_Warning_Limit (CAN_SFRmap* CANx)
功能描述	获取 CAN 总线错误信息
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
返回值	错误报警限制, 取值为 8 位数值
被调用函数	无

6.4.17 函数 CAN_Get_Error_Counter

表 6-21 函数 CAN_Get_Error_Counter

函数名	CAN_Get_Error_Counter
函数原型	uint8_t CAN_Get_Error_Counter (CAN_SFRmap* CANx, uint32_t Direction)
功能描述	获取 CAN 发送/接收错误计数器
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Direction: CAN 错误传输方向, 取值为: CAN_ERROR_AT_TX: 发送错误计数器 CAN_ERROR_AT_RX: 接收错误计数器
返回值	错误报警限制, 取值为 8 位数值
被调用函数	无

6.4.18 函数 CAN_Error_Warning_Limit_Config

表 6-22 函数 CAN_Error_Warning_Limit_Config

函数名	CAN_Error_Warning_Limit_Config
函数原型	void CAN_Error_Warning_Limit_Config (CAN_SFRmap* CANx, uint8_t ErrorLimit)
功能描述	配置 CAN 错误报警限制
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	ErrorLimit: 错误报警限制, 取值为 8 位有效数值
返回值	错误报警限制, 取值为 8 位数值
被调用函数	无

6.4.19 函数 CAN_Error_Counter_Config

表 6-23 函数 CAN_Error_Counter_Config

函数名	CAN_Error_Counter_Config
函数原型	void CAN_Error_Counter_Config (CAN_SFRmap* CANx, uint32_t Direction, uint8_t ErrorCounter)
功能描述	配置 CAN 发送/接收错误计数器
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Direction: CAN 错误传输方向, 取值为: CAN_ERROR_AT_TX: 发送错误计数器 CAN_ERROR_AT_RX: 接收错误计数器
输入参数 3	错误计数, 取值为 8 位有效数值
返回值	无
被调用函数	无

6.4.20 函数 CAN_Acceptance_Config

表 6-24 函数 CAN_Acceptance_Config

函数名	CAN_Acceptance_Config
函数原型	void CAN_Acceptance_Config (CAN_SFRmap* CANx, uint32_t Acceptance)
功能描述	配置 CAN 验收代码
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Acceptance: 验收代码, 取值为 32 位有效数值
返回值	无
被调用函数	无

6.4.21 函数 CAN_Get_Acceptance

表 6-25 函数 CAN_Acceptance_Config

函数名	CAN_Acceptance_Config
函数原型	uint32_t CAN_Get_Acceptance (CAN_SFRmap* CANx)
功能描述	获取 CAN 验收代码
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
返回值	验收代码, 取值为 32 位有效数值
被调用函数	无

6.4.22 函数 CAN_Acceptance_Mask_Config

表 6-26 函数 CAN_Acceptance_Mask_Config

函数名	CAN_Acceptance_Mask_Config
-----	----------------------------

函数原型	void CAN_Acceptance_Mask_Config (CAN_SFRmap* CANx, uint32_t Acceptance)
功能描述	配置 CAN 验收屏蔽
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Acceptance: 验收屏蔽, 取值为 32 位有效数值
返回值	无
被调用函数	无

6. 4. 23 函数 CAN_Get_Acceptance_Mask

表 6-27 函数 CAN_Get_Acceptance_Mask

函数名	CAN_Get_Acceptance_Mask
函数原型	uint32_t CAN_Get_Acceptance_Mask (CAN_SFRmap* CANx)
功能描述	获取 CAN 验收屏蔽码
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
返回值	验收屏蔽代码, 取值为 32 位有效数值
被调用函数	无

6. 4. 24 函数 CAN_Transmit_Message_Configuration

表 6-28 函数 CAN_Transmit_Message_Configuration

函数名	CAN_Transmit_Message_Configuration
函数原型	void CAN_Transmit_Message_Configuration (CAN_SFRmap* CANx, CAN_MessageTypeDef* canInitStruct)
功能描述	CAN 模块(CAN)发送缓冲配置
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	canInitStruct: CAN 模块配置信息结构体指针
返回值	无
被调用函数	无

6. 4. 25 函数 CAN_Receive_Message_Configuration

表 6-29 函数 CAN_Receive_Message_Configuration

函数名	CAN_Receive_Message_Configuration
函数原型	void CAN_Receive_Message_Configuration (CAN_SFRmap* CANx, uint32_t ReceiveOffset, CAN_MessageTypeDef* canInitStruct)
功能描述	获取 CAN 模块(CAN)接收缓冲的一个帧信息
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	ReceiveOffset: RAM 存储空间起始地址, 取值为 8 位有效数, 4 字节对齐
输入参数 3	canInitStruct: CAN 模块配置信息结构体指针

返回值	无
被调用函数	无

6. 4. 26 函数 CAN_Message_Struct_Init

表 6-30 函数 CAN_Message_Struct_Init

函数名	CAN_Message_Struct_Init
函数原型	void CAN_Message_Struct_Init (CAN_MessageTypeDef* canInitStruct)
功能描述	初始化 CAN 报文信息结构体
输入参数 1	canInitStruct: CAN 模块配置信息结构体指针
返回值	无
被调用函数	无

6. 4. 27 函数 CAN_Clear_Buffer_Overflow_Flag

表 6-31 函数 CAN_Clear_Buffer_Overflow_Flag

函数名	CAN_Clear_Buffer_Overflow_Flag
函数原型	void CAN_Clear_Buffer_Overflow_Flag (CAN_SFRmap* CANx)
功能描述	清除 CAN 清除缓冲器满标志
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
返回值	无
被调用函数	无

6. 4. 28 函数 CAN_Release_Receive_Buffer

表 6-32 函数 CAN_Release_Receive_Buffer

函数名	CAN_Release_Receive_Buffer
函数原型	void CAN_Release_Receive_Buffer (CAN_SFRmap* CANx, uint32_t ReleaseCount)
功能描述	释放 CAN 接收缓冲器
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	ReleaseCount: 释放缓冲器次数
返回值	无
被调用函数	无

6. 4. 29 函数 CAN_Transmit_Single

表 6-33 函数 CAN_Transmit_Single

函数名	CAN_Transmit_Single
-----	---------------------

函数原型	void CAN_Transmit_Single (CAN_SFRmap* CANx)
功能描述	单次发送，当发生错误或者仲裁丢失时不会进行重发（单次发送）
输入参数 1	CANx: 指向 CAN 内存结构的指针，取值为 CAN0_SFR~CAN5_SFR
返回值	无
被调用函数	无

6. 4. 30 函数 CAN_Transmit_Repeat

表 6-34 函数 CAN_Transmit_Repeat

函数名	CAN_Transmit_Repeat
函数原型	void CAN_Transmit_Repeat (CAN_SFRmap* CANx)
功能描述	发送，发生错误、仲裁丢失或发送结束时会进行重发（连续发送）
输入参数 1	CANx: 指向 CAN 内存结构的指针，取值为 CAN0_SFR~CAN5_SFR
返回值	无
被调用函数	无

6. 4. 31 函数 CAN_Frame_Format_Config

表 6-35 函数 CAN_Frame_Format_Config

函数名	CAN_Frame_Format_Config
函数原型	void CAN_Frame_Format_Config (CAN_SFRmap* CANx, uint32_t FrameFormat)
功能描述	配置 CAN 发送缓冲器帧格式
输入参数 1	CANx: 指向 CAN 内存结构的指针，取值为 CAN0_SFR~CAN5_SFR
输入参数 2	FrameFormat: 帧格式，取值为: CAN_FRAME_FORMAT_SFF: 标准帧格式 SFF CAN_FRAME_FORMAT_EFF: 扩展帧格式 EFF
返回值	无
被调用函数	无

6. 4. 32 函数 CAN_Remote_Request_Config

表 6-36 函数 CAN_Remote_Request_Config

函数名	CAN_Remote_Request_Config
函数原型	void CAN_Remote_Request_Config (CAN_SFRmap* CANx, uint32_t RemoteRequest)
功能描述	配置 CAN 发送缓冲器远程发送请求
输入参数 1	CANx: 指向 CAN 内存结构的指针，取值为 CAN0_SFR~CAN5_SFR
输入参数 2	FrameFormat: 帧格式，取值为: CAN_FRAME_FORMAT_SFF: 标准帧格式 SFF

	CAN_FRAME_FORMAT_EFF: 扩展帧格式 EFF
返回值	无
被调用函数	无

6.4.33 函数 CAN_Data_Length_Config

表 6-37 函数 CAN_Data_Length_Config

函数名	CAN_Data_Length_Config
函数原型	void CAN_Data_Length_Config (CAN_SFRmap* CANx, uint32_t Length)
功能描述	配置 CAN 发送缓冲器数据长度, 大于 8 的数据长度代码是不可用的; 如果大于 8, 将以 8 个字节计
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Length: CAN 数据长度, 取值为 4 位有效数值。如果大于 8, 将以 8 个字节计
返回值	无
被调用函数	无

6.4.34 函数 CAN_Identification_Code_Config

表 6-38 函数 CAN_Identification_Code_Config

函数名	CAN_Identification_Code_Config
函数原型	void CAN_Identification_Code_Config (CAN_SFRmap* CANx, uint32_t FrameFormat, uint32_t IDCode)
功能描述	配置 CAN 发送缓冲器识别码
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	Format: 帧格式, 取值为: CAN_FRAME_FORMAT_SFF: 标准帧格式 SFF CAN_FRAME_FORMAT_EFF: 扩展帧格式 EFF
返回值	无
被调用函数	无

6.4.35 函数 CAN_Get_INT_Flag

表 6-39 函数 CAN_Get_INT_Flag

函数名	CAN_Get_INT_Flag
函数原型	FlagStatus CAN_Get_INT_Flag (CAN_SFRmap* CANx, uint32_t InterruptType)
功能描述	获取 CAN 中断标志
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	InterruptType: CAN 中断类型, 取值范围为下面提供的一个:

	CAN_INT_RECEIVE: 接收中断 CAN_INT_TRANSMIT: 发送中断 CAN_INT_ERROR_ALARM: 错误报警中断 CAN_INT_DATA_OVERFLOW: 数据溢出中断 CAN_INT_WAKE_UP: 唤醒中断 CAN_INT_ERROR_NEGATIVE: 错误消极中断 CAN_INT_ARBITRATION_LOST: 仲裁丢失中断 CAN_INT_BUS_ERROR: 总线错误中断
返回值	1:发生中断, 0:未发生中断
被调用函数	无

6.4.36 函数 CAN_Clear_INT_Flag

表 6-40 函数 CAN_Clear_INT_Flag

函数名	CAN_Clear_INT_Flag
函数原型	void CAN_Clear_INT_Flag (CAN_SFRmap* CANx, uint32_t InterruptType)
功能描述	清零 CAN 中断标志
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	InterruptType: CAN 中断类型, 取值范围为下面提供的一个: CAN_INT_RECEIVE: 接收中断 CAN_INT_TRANSMIT: 发送中断 CAN_INT_ERROR_ALARM: 错误报警中断 CAN_INT_DATA_OVERFLOW: 数据溢出中断 CAN_INT_WAKE_UP: 唤醒中断 CAN_INT_ERROR_NEGATIVE: 错误消极中断 CAN_INT_ARBITRATION_LOST: 仲裁丢失中断 CAN_INT_BUS_ERROR: 总线错误中断
返回值	无
被调用函数	无

6.4.37 函数 CAN_Set_INT_Enable

表 6-41 函数 CAN_Set_INT_Enable

函数名	CAN_Set_INT_Enable
函数原型	void CAN_Set_INT_Enable (CAN_SFRmap* CANx, uint32_t InterruptType, FunctionalState NewState)
功能描述	配置 CAN 中断使能
输入参数 1	CANx: 指向 CAN 内存结构的指针, 取值为 CAN0_SFR~CAN5_SFR
输入参数 2	InterruptType: CAN 中断类型, 取值范围为下面提供的一个: CAN_INT_RECEIVE: 接收中断 CAN_INT_TRANSMIT: 发送中断

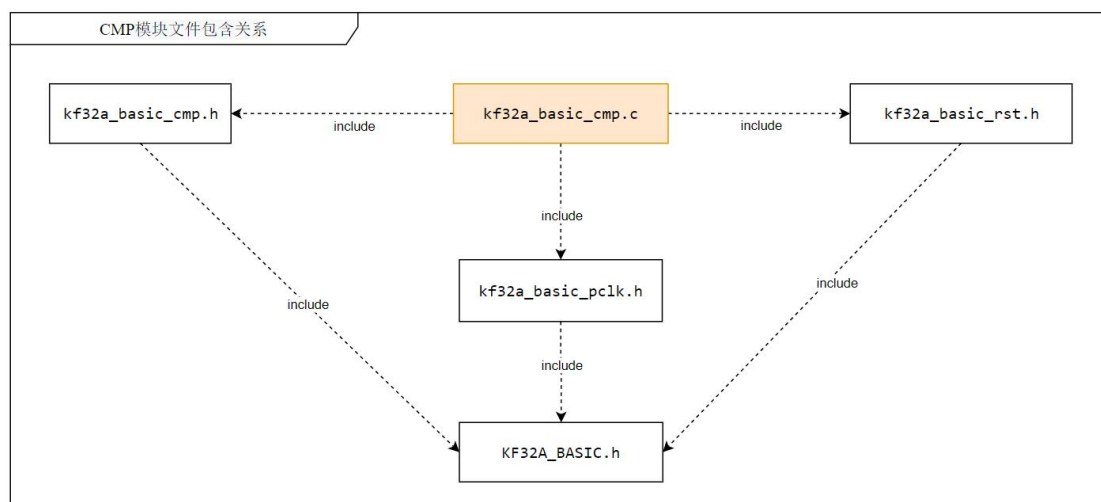
	CAN_INT_ERROR_ALARM: 错误报警中断 CAN_INT_DATA_OVERFLOW: 数据溢出中断 CAN_INT_WAKE_UP: 唤醒中断 CAN_INT_ERROR_NEGATIVE: 错误消极中断 CAN_INT_ARBITRATION_LOST: 仲裁丢失中断 CAN_INT_BUS_ERROR: 总线错误中断
输入参数 3	NewState: DMA 通道错误传输中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

7 模拟比较器模块(CMP)

模拟比较器模块（CMP）主要特点正负端多输入端口可选;电阻分压模块提供可选内部参考电压;输出极性可选;中断边沿可选;数字滤波功能;比较器输出可作为定时器捕捉输入、PWM 关断源或用于清零定时器;可配置为 BEMF（反向电动势）模式和 HALL（霍尔检测）模式。

7.1 文件引用关系

图 7-1 CMP 模块文件包含关系



7.2 CMP 寄存器结构

CMP 寄存器结构，CMP_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct CMP_MemMap {
    volatile uint32_t    CTL;
    volatile uint32_t    CTL1;
    volatile uint32_t    CTL2;
    volatile uint32_t    CTL3;
    volatile uint32_t    CTL4;
}CMP_SFRmap;
    
```

表 7-1CMP 寄存器结构说明

寄存器	描述
CTL	CMP 控制寄存器 0, 偏移:0x0
CTL1	CMP 控制寄存器 1, 偏移:0x4
CTL2	CMP 控制寄存器 2, 偏移:0x8
CTL3	CMP 控制寄存器 3, 偏移:0xC

CTL4	CMP 控制寄存器 4, 偏移:0x10
------	----------------------

CMP 配置信息结构体, CMP_SFRmap, 定义于文件 kf32a_basic_cmp.h 中, 如下:

```
typedef struct
{
    uint32_t    m_PositiveInput;
    uint32_t    m_NegativeInput;
    FunctionalState    m_FallTriggerEnable;
    FunctionalState    m_RiseTriggerEnable;
    uint32_t    m_Clock;
    uint32_t    m_FrequencyDivision;
    uint32_t    m_SampleNumber;
    FunctionalState    m_FilterEnable;
    FunctionalState    m_ScopecontrolEnable;
    uint32_t    m_OutputPolarity;
    FunctionalState    m_CmpEnable;
}CMP_InitTypeDef;//CMP0/CMP1/CMP2/CMP3
```

表 7-2 CMP 配置信息结构体说明

寄存器	描述
m_PositiveInput	CMP 的正端输出选择位,取值为宏“CMP 正端输入选择”中的一个
m_NegativeInput	CMP 的负端输出选择位,取值为宏“CMP 负端输入选择”中的一个
m_FallTriggerEnable	CMP 下降沿触发中断使能, 只有 CMP0/CMP1/CMP2,取值为 TRUE 或 FALSE
m_RiseTriggerEnable	CMP 上升沿触发中断使能.只有 CMP0/CMP1/CMP2, 取值为 TRUE 或 FALSE
m_Clock	CMP 滤波器滤波时钟源,取值为宏“CMP 滤波器滤波时钟源选择位”中的一个
m_FrequencyDivision	CMP 滤波器滤波时钟分频,取值为 8 位数据, 0-255 中的一个
m_SampleNumber	CMP 取样数量选择, 取值为宏“CMP 滤波器取样数量选择”中的一个
m_FilterEnable	CMP 滤波器使能位, ,取值为 TRUE 或 FALSE
m_ScopecontrolEnable	CMP 范围控制使能位, ,取值为 TRUE 或 FALSE
m_OutputPolarity	CMP 输出极性选择,取值为宏“CMP 输出极性选择”中的一个
m_CmpEnable	CMP 比较器使能, 取值为 TRUE 或 FALSE

7.3 CMP 宏定义

CMP 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, CMP 其他相关宏定义详见文件 kf32a_basic_cmp.h 及函数参数描述。

7.4 CMP 库函数

表 7-4 CMP 固件库函数列表

序号	函数名	描述
1	CMP_Reset	CMP 外设复位，使能外设时钟
2	CMP_Configuration	CMP 外设配置
3	CMP_Struct_Init	初始化 CMP 配置信息结构体
4	CMP0_Cmd	控制 CMP0 使能位
5	CMP1_Cmd	控制 CMP1 使能位
6	CMP2_Cmd	控制 CMP2 使能位
7	CMP3_Cmd	控制 CMP3 使能位
8	CMP0_POSITIVE_INPUT_SELECT	CMP0 正端输入端口信号选择
9	CMP0_NEGATIVE_INPUT_SELECT	CMP0 负端输入端口信号选择
10	CMP1_POSITIVE_INPUT_SELECT	CMP1 正端输入端口信号选择
11	CMP1_NEGATIVE_INPUT_SELECT	CMP1 负端输入端口信号选择
12	CMP2_POSITIVE_INPUT_SELECT	CMP2 正端输入端口信号选择
13	CMP2_NEGATIVE_INPUT_SELECT	CMP2 负端输入端口信号选择
14	CMP3_POSITIVE_INPUT_SELECT	CMP3 正端输入端口信号选择
15	CMP3_NEGATIVE_INPUT_SELECT	CMP3 负端输入端口信号选择
16	CMP0_OUTPUT_POL_SELECT	CMP0 输出极性选择
17	CMP1_OUTPUT_POL_SELECT	CMP1 输出极性选择
18	CMP2_OUTPUT_POL_SELECT	CMP2 输出极性选择
19	CMP3_OUTPUT_POL_SELECT	CMP3 输出极性选择
20	CMP_OUTPUT_SELECT	CMP 输出选择
21	CMP0_Read_Output_State	读 CMP0 输出数据
22	CMP1_Read_Output_State	读 CMP1 输出数据
23	CMP2_Read_Output_State	读 CMP2 输出数据
24	CMP3_Read_Output_State	读 CMP3 输出数据
25	CMP0_Get_Updata_INT_Flag	读 CMP0 中断标志
26	CMP1_Get_Updata_INT_Flag	读 CMP1 中断标志
27	CMP2_Get_Updata_INT_Flag	读 CMP2 中断标志
28	CMP3_Get_Updata_INT_Flag	读 CMP3 中断标志
29	CMP_Trigger_Select_Config	配置触发选择位
30	CMP0_Clear_Trigger_INT_Flag	清除比较器 0 中断标志
31	CMP1_Clear_Trigger_INT_Flag	清除比较器 1 中断标志
32	CMP2_Clear_Trigger_INT_Flag	清除比较器 2 中断标志
33	CMP3_Clear_Trigger_INT_Flag	清除比较器 3 中断标志
34	CMP0_INT_Enable	配置 CMP0 中断使能
35	CMP1_INT_Enable	配置 CMP1 中断使能
36	CMP2_INT_Enable	配置 CMP2 中断使能

37	CMP3_INT_Enable	配置 CMP3 中断使能
38	CMP_SluggishVoltage_Select	配置 CMP 迟滞电压选择位
39	CMP_HALLMODE_Select	霍尔模式选择位
40	CMP_BEMF_Enable	反向电动势模式使能
41	CMP_FLTINSEL_Select	滤波器输入选择位

7.4.1 函数 CMP_Reset

表 7-2 函数 CMP_Reset

函数名	CMP_Reset
函数原型	void CMP_Reset ()
功能描述	CMP 外设复位，使能外设时钟
返回值	无
被调用函数	RST_CTL1_Peripheral_Reset_Enable(RST_CTL1_CMPRST, TRUE); RST_CTL1_Peripheral_Reset_Enable(RST_CTL1_CMPRST, FALSE); PCLK_CTL1_Peripheral_Clock_Enable(PCLK_CTL1_CMPCLKEN, TRUE);

7.4.2 函数 CMP_Configuration

表 7-3 函数 CMP_Configuration

函数名	CMP_Configuration
函数原型	void CMP_Configuration (CMP_SFRmap * CMPx, CMP_InitTypeDef* CMPInitStruct)
功能描述	CMP 外设配置
输入参数 1	CMPx: 取值为 CMP0_SFR、CMP1_SFR、CMP2_SFR、CMP3_SFR
输入参数 2	CMPInitStruct: CMP 配置信息
返回值	无
被调用函数	无

7.4.3 函数 CMP_Struct_Init

表 7-4 函数 CMP_Struct_Init

函数名	CMP_Struct_Init
函数原型	void CMP_Struct_Init (CMP_InitTypeDef* CMPInitStruct)
功能描述	初始化 CMP 配置信息结构体
输入参数 1	CMPInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

7.4.4 函数 CMP0_Cmd

表 7-5 函数 CMP0_Cmd

函数名	CMP0_Cmd
函数原型	void CMP0_Cmd (FunctionalState NewState)
功能描述	控制 CMP0 使能位
输入参数 1	NewState: CMP 使能位配置信息, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.5 函数 CMP1_Cmd

表 7-6 函数 CMP1_Cmd

函数名	CMP1_Cmd
函数原型	void CMP1_Cmd (FunctionalState NewState)
功能描述	控制 CMP1 使能位
输入参数 1	NewState: CMP 使能位配置信息, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.6 函数 CMP2_Cmd

表 7-7 函数 CMP2_Cmd

函数名	CMP2_Cmd
函数原型	void CMP2_Cmd (FunctionalState NewState)
功能描述	控制 CMP2 使能位
输入参数 1	NewState: CMP 使能位配置信息, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.7 函数 CMP3_Cmd

表 7-8 函数 CMP3_Cmd

函数名	CMP3_Cmd
函数原型	void CMP3_Cmd (FunctionalState NewState)
功能描述	控制 CMP3 使能位
输入参数 1	NewState: CMP 使能位配置信息, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.8 函数 CMP0_POSITIVE_INPUT_SELECT

表 7-9 函数 CMP0_POSITIVE_INPUT_SELECT

函数名	CMP0_POSITIVE_INPUT_SELECT
函数原型	void CMP0_POSITIVE_INPUT_SELECT (uint32_t Select)
功能描述	CMP0 正端输入端口信号选择
输入参数 1	CMP0_PositiveINPUT_PIN_PA0 CMP0_PositiveINPUT_PIN_PA9 CMP0_PositiveINPUT_PIN_PB2 CMP0_PositiveINPUT_PIN_PB9 CMP0_PositiveINPUT_PIN_PC11 CMP0_PositiveINPUT_PIN_AGND CMP0_PositiveINPUT_PIN_DAC0OUT CMP0_PositiveINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.9 函数 CMP0_NEGATIVE_INPUT_SELECT

表 7-10 函数 CMP0_NEGATIVE_INPUT_SELECT

函数名	CMP0_NEGATIVE_INPUT_SELECT
函数原型	void CMP0_NEGATIVE_INPUT_SELECT (uint32_t Select)
功能描述	CMP0 负端输入端口信号选择
输入参数 1	CMP0_NegativeINPUT_PIN_PA1 CMP0_NegativeINPUT_PIN_PA10 CMP0_NegativeINPUT_PIN_PB3 CMP0_NegativeINPUT_PIN_PB10 CMP0_NegativeINPUT_PIN_PC12 CMP0_NegativeINPUT_PIN_AGND CMP0_NegativeINPUT_PIN_DAC0OUT CMP0_NegativeINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.10 函数 CMP1_POSITIVE_INPUT_SELECT

表 7-11 函数 CMP1_POSITIVE_INPUT_SELECT

函数名	CMP1_POSITIVE_INPUT_SELECT
函数原型	void CMP1_POSITIVE_INPUT_SELECT (uint32_t Select)
功能描述	CMP1 正端输入端口信号选择
输入参数 1	CMP1_PositiveINPUT_PIN_PA0

	CMP1_PositiveINPUT_PIN_PA9 CMP1_PositiveINPUT_PIN_PB2 CMP1_PositiveINPUT_PIN_PB11 CMP1_PositiveINPUT_PIN_PG7 CMP1_PositiveINPUT_PIN_AGND CMP1_PositiveINPUT_PIN_DAC0OUT CMP1_PositiveINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.11 函数 CMP1_NEGATIVE_INPUT_SELECT

表 7-12 函数 CMP1_NEGATIVE_INPUT_SELECT

函数名	CMP1_NEGATIVE_INPUT_SELECT
函数原型	void CMP1_NEGATIVE_INPUT_SELECT (uint31_t Select)
功能描述	CMP1 负端输入端口信号选择
输入参数 1	CMP1_NegativeINPUT_PIN_PA1 CMP1_NegativeINPUT_PIN_PA10 CMP1_NegativeINPUT_PIN_PB3 CMP1_NegativeINPUT_PIN_PB12 CMP1_NegativeINPUT_PIN_PC9 CMP1_NegativeINPUT_PIN_AGND CMP1_NegativeINPUT_PIN_DAC0OUT CMP1_NegativeINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.12 函数 CMP2_POSITIVE_INPUT_SELECT

表 7-13 函数 CMP2_POSITIVE_INPUT_SELECT

函数名	CMP2_POSITIVE_INPUT_SELECT
函数原型	void CMP2_POSITIVE_INPUT_SELECT (uint22_t Select)
功能描述	CMP2 正端输入端口信号选择
输入参数 1	CMP2_PositiveINPUT_PIN_PA0 CMP2_PositiveINPUT_PIN_PA9 CMP2_PositiveINPUT_PIN_PB2 CMP2_PositiveINPUT_PIN_PB13 CMP2_PositiveINPUT_PIN_PC7 CMP2_PositiveINPUT_PIN_AGND CMP2_PositiveINPUT_PIN_DAC0OUT CMP2_PositiveINPUT_PIN_DAC1OUT

返回值	无
被调用函数	无

7.4.13 函数 CMP2_NEGATIVE_INPUT_SELECT

表 7-14 函数 CMP2_NEGATIVE_INPUT_SELECT

函数名	CMP2_NEGATIVE_INPUT_SELECT
函数原型	void CMP2_NEGATIVE_INPUT_SELECT (uint22_t Select)
功能描述	CMP2 负端输入端口信号选择
输入参数 1	CMP2_NegativeINPUT_PIN_PA1 CMP2_NegativeINPUT_PIN_PA10 CMP2_NegativeINPUT_PIN_PB3 CMP2_NegativeINPUT_PIN_PB14 CMP2_NegativeINPUT_PIN_PC8 CMP2_NegativeINPUT_PIN_AGND CMP2_NegativeINPUT_PIN_DAC0OUT CMP2_NegativeINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.14 函数 CMP3_POSITIVE_INPUT_SELECT

表 7-15 函数 CMP3_POSITIVE_INPUT_SELECT

函数名	CMP3_POSITIVE_INPUT_SELECT
函数原型	void CMP3_POSITIVE_INPUT_SELECT (uint22_t Select)
功能描述	CMP3 正端输入端口信号选择
输入参数 1	CMP3_PositiveINPUT_PIN_OP2OUT CMP3_PositiveINPUT_PIN_PA0 CMP3_PositiveINPUT_PIN_PA9 CMP3_PositiveINPUT_PIN_PB2 CMP3_PositiveINPUT_PIN_PB15 CMP3_PositiveINPUT_PIN_PC5 CMP3_PositiveINPUT_PIN_AGND CMP3_PositiveINPUT_PIN_DAC0OUT CMP3_PositiveINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.15 函数 CMP3_NEGATIVE_INPUT_SELECT

表 7-16 函数 CMP3_NEGATIVE_INPUT_SELECT

函数名	CMP3_NEGATIVE_INPUT_SELECT
函数原型	void CMP3_NEGATIVE_INPUT_SELECT (uint22_t Select)
功能描述	CMP3 负端输入端口信号选择
输入参数 1	CMP3_NegativeINPUT_PIN_PA1 CMP3_NegativeINPUT_PIN_PA10 CMP3_NegativeINPUT_PIN_PB3 CMP3_NegativeINPUT_PIN_PF0 CMP3_NegativeINPUT_PIN_PC6 CMP3_NegativeINPUT_PIN_AGND CMP3_NegativeINPUT_PIN_DAC0OUT CMP3_NegativeINPUT_PIN_DAC1OUT
返回值	无
被调用函数	无

7.4.16 函数 CMP0_OUTPUT_POL_SELECT

表 7-17 函数 CMP0_OUTPUT_POL_SELECT

函数名	CMP0_OUTPUT_POL_SELECT
函数原型	void CMP0_OUTPUT_POL_SELECT (uint32_t Select)
功能描述	CMP0 输出极性选择
输入参数 1	CMP0_OUTPUT_Normal CMP0_OUTPUT_Opposite
返回值	无
被调用函数	无

7.4.17 函数 CMP1_OUTPUT_POL_SELECT

表 7-18 函数 CMP1_OUTPUT_POL_SELECT

函数名	CMP1_OUTPUT_POL_SELECT
函数原型	void CMP1_OUTPUT_POL_SELECT (uint32_t Select)
功能描述	CMP1 输出极性选择
输入参数 1	CMP1_OUTPUT_Normal CMP1_OUTPUT_Opposite
返回值	无
被调用函数	无

7.4.18 函数 CMP2_OUTPUT_POL_SELECT

表 7-19 函数 CMP2_OUTPUT_POL_SELECT

函数名	CMP2_OUTPUT_POL_SELECT
函数原型	void CMP2_OUTPUT_POL_SELECT (uint32_t Select)
功能描述	CMP2 输出极性选择
输入参数 1	CMP2_OUTPUT_Normal CMP2_OUTPUT_Opposite
返回值	无
被调用函数	无

7.4.19 函数 CMP3_OUTPUT_POL_SELECT

表 7-20 函数 CMP3_OUTPUT_POL_SELECT

函数名	CMP3_OUTPUT_POL_SELECT
函数原型	void CMP3_OUTPUT_POL_SELECT (uint32_t Select)
功能描述	CMP3 输出极性选择
输入参数 1	CMP3_OUTPUT_Normal CMP3_OUTPUT_Opposite
返回值	无
被调用函数	无

7.4.20 函数 CMP_OUTPUT_SELECT

表 7-21 函数 CMP_OUTPUT_SELECT

函数名	CMP_OUTPUT_SELECT
函数原型	void CMP_OUTPUT_SELECT (uint32_t Select)
功能描述	CMP 输出选择
输入参数 1	NONE CMP0_OUTPUT CMP1_OUTPUT CMP2_OUTPUT
返回值	无
被调用函数	无

7.4.21 函数 CMP0_Read_Output_State

表 7-22 函数 CMP0_Read_Output_State

函数名	CMP0_Read_Output_State
函数原型	FlagStatus CMP0_Read_Output_State ()

功能描述	读 CMP0 输出数据
返回值	比较器输出状态
被调用函数	无

7.4.22 函数 CMP1_Read_Output_State

表 7-23 函数 CMP1_Read_Output_State

函数名	CMP1_Read_Output_State
函数原型	FlagStatus CMP1_Read_Output_State ()
功能描述	读 CMP1 输出数据
返回值	比较器输出状态
被调用函数	无

7.4.23 函数 CMP2_Read_Output_State

表 7-24 函数 CMP2_Read_Output_State

函数名	CMP2_Read_Output_State
函数原型	FlagStatus CMP2_Read_Output_State ()
功能描述	读 CMP2 输出数据
返回值	比较器输出状态
被调用函数	无

7.4.24 函数 CMP3_Read_Output_State

表 7-25 函数 CMP3_Read_Output_State

函数名	CMP3_Read_Output_State
函数原型	FlagStatus CMP3_Read_Output_State ()
功能描述	读 CMP3 输出数据
返回值	比较器输出状态
被调用函数	无

7.4.25 函数 CMP0_Get_Updata_INT_Flag

表 7-26 函数 CMP0_Get_Updata_INT_Flag

函数名	CMP0_Get_Updata_INT_Flag
函数原型	FlagStatus CMP0_Get_Updata_INT_Flag ()
功能描述	读 CMP0 中断标志
返回值	比较器中断标志状态
被调用函数	无

7. 4. 26 函数 CMP1_Get_Updata_INT_Flag

表 7-27 函数 CMP1_Get_Updata_INT_Flag

函数名	CMP1_Get_Updata_INT_Flag
函数原型	FlagStatus CMP1_Get_Updata_INT_Flag ()
功能描述	读 CMP1 中断标志
返回值	比较器中断标志状态
被调用函数	无

7. 4. 27 函数 CMP2_Get_Updata_INT_Flag

表 7-28 函数 CMP2_Get_Updata_INT_Flag

函数名	CMP2_Get_Updata_INT_Flag
函数原型	FlagStatus CMP2_Get_Updata_INT_Flag ()
功能描述	读 CMP2 中断标志
返回值	比较器中断标志状态
被调用函数	无

7. 4. 28 函数 CMP3_Get_Updata_INT_Flag

表 7-29 函数 CMP3_Get_Updata_INT_Flag

函数名	CMP3_Get_Updata_INT_Flag
函数原型	FlagStatus CMP3_Get_Updata_INT_Flag ()
功能描述	读 CMP3 中断标志
返回值	比较器中断标志状态
被调用函数	无

7. 4. 29 函数 CMP_Trigger_Select_Config

表 7-30 函数 CMP_Trigger_Select_Config

函数名	CMP_Trigger_Select_Config
函数原型	void CMP_Trigger_Select_Config (uint32_t TriggerSelect)
功能描述	配置触发选择位
输入参数 1	TriggerSelect: 触发选择, 取值范围为: CMP_CMPOUT_FlipLatch_INT CMP_CMPOUT_Change_INT
返回值	无
被调用函数	无

7.4.30 函数 CMP0_Clear_Trigger_INT_Flag

表 7-31 函数 CMP0_Clear_Trigger_INT_Flag

函数名	CMP0_Clear_Trigger_INT_Flag
函数原型	void CMP0_Clear_Trigger_INT_Flag ()
功能描述	清除比较器 0 中断标志
返回值	无
被调用函数	无

7.4.31 函数 CMP1_Clear_Trigger_INT_Flag

表 7-32 函数 CMP1_Clear_Trigger_INT_Flag

函数名	CMP1_Clear_Trigger_INT_Flag
函数原型	void CMP1_Clear_Trigger_INT_Flag ()
功能描述	清除比较器 1 中断标志
返回值	无
被调用函数	无

7.4.32 函数 CMP2_Clear_Trigger_INT_Flag

表 7-33 函数 CMP2_Clear_Trigger_INT_Flag

函数名	CMP2_Clear_Trigger_INT_Flag
函数原型	void CMP2_Clear_Trigger_INT_Flag ()
功能描述	清除比较器 2 中断标志
返回值	无
被调用函数	无

7.4.33 函数 CMP3_Clear_Trigger_INT_Flag

表 7-34 函数 CMP3_Clear_Trigger_INT_Flag

函数名	CMP3_Clear_Trigger_INT_Flag
函数原型	void CMP3_Clear_Trigger_INT_Flag ()
功能描述	清除比较器 3 中断标志
返回值	无
被调用函数	无

7.4.34 函数 CMP0_INT_Enable

表 7-35 函数 CMP0_INT_Enable

函数名	CMP0_INT_Enable
-----	-----------------

函数原型	void CMP0_INT_Enable (FunctionalState NewState)
功能描述	配置 CMP0 中断使能
输入参数 1	NewState: 中断取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.35 函数 CMP1_INT_Enable

表 7-36 函数 CMP1_INT_Enable

函数名	CMP1_INT_Enable
函数原型	void CMP1_INT_Enable (FunctionalState NewState)
功能描述	配置 CMP1 中断使能
输入参数 1	NewState: 中断取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.36 函数 CMP2_INT_Enable

表 7-37 函数 CMP2_INT_Enable

函数名	CMP2_INT_Enable
函数原型	void CMP2_INT_Enable (FunctionalState NewState)
功能描述	配置 CMP2 中断使能
输入参数 1	NewState: 中断取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.37 函数 CMP3_INT_Enable

表 7-38 函数 CMP3_INT_Enable

函数名	CMP3_INT_Enable
函数原型	void CMP3_INT_Enable (FunctionalState NewState)
功能描述	配置 CMP3 中断使能
输入参数 1	NewState: 中断取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.38 函数 CMP_SluggishVoltage_Select

表 7-39 函数 CMP_SluggishVoltage_Select

函数名	CMP_SluggishVoltage_Select
函数原型	void CMP_SluggishVoltage_Select (uint32_t Select)
功能描述	配置 CMP 迟滞电压选择位
输入参数 1	Select: 取值范围为: CMP_SluggishVoltage_OFF CMP_SluggishVoltage_5mV CMP_SluggishVoltage_10mV CMP_SluggishVoltage_15mV
返回值	无
被调用函数	无

7.4.39 函数 CMP_HALLMODE_Select

表 7-40 函数 CMP_HALLMODE_Select

函数名	CMP_HALLMODE_Select
函数原型	void CMP_HALLMODE_Select (uint32_t Select)
功能描述	霍尔模式选择位
输入参数 1	Select: 取值范围为: CMP_HALLMODE_SINGLE CMP_HALLMODE_BOTH
返回值	无
被调用函数	无

7.4.40 函数 CMP_BEMF_Enable

表 7-41 函数 CMP_BEMF_Enable

函数名	CMP_BEMF_Enable
函数原型	void CMP_BEMF_Enable (FunctionalState NewState)
功能描述	反向电动势模式使能
输入参数 1	NewState: 中断取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

7.4.41 函数 CMP_FLTINSEL_Select

表 7-42 函数 CMP_FLTINSEL_Select

函数名	CMP_FLTINSEL_Select
-----	---------------------

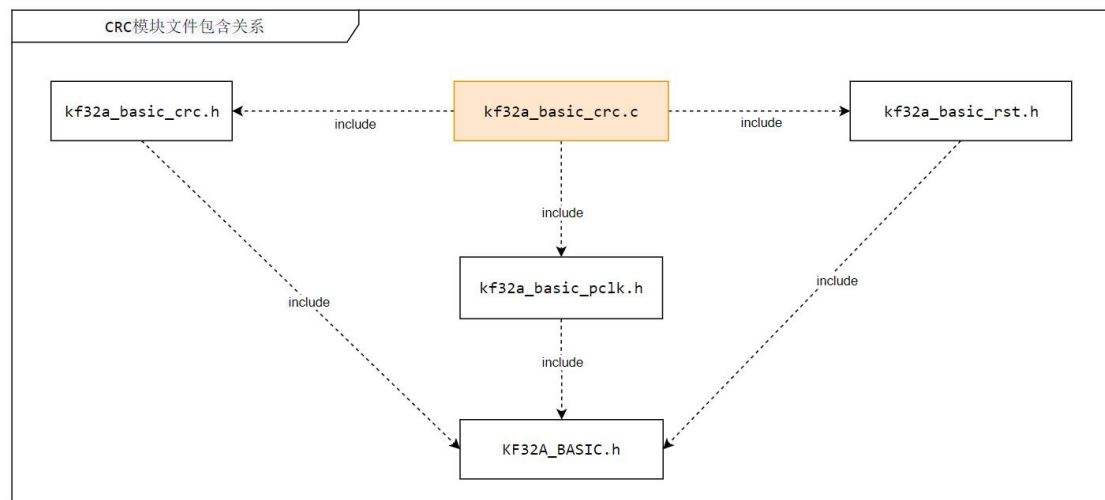
函数原型	void CMP_FLTINSEL_Select (uint32_t Select)
功能描述	滤波器输入选择位
输入参数 1	Select: 取值范围为: CMP_FLTINSEL_CMPOUT CMP_FLTINSEL_IO
返回值	无
被调用函数	无

8 循环冗余校验单元（CRC）

循环冗余校验单元（Cyclic Redundancy Check, CRC）可以通过生成多项式计算不同长度数据的 CRC 校验值。CRC 技术可应用于核实数据传输或者数据存储的正确性和完整性。

8.1 文件引用关系

图 8-1 CRC 模块文件包含关系



8.2 CRC 寄存器结构

CRC 寄存器结构，CRC_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct CRC_MemMap {
    volatile uint32_t    CTL;
    Volatile uint32_t    DATA;
    Volatile uint32_t    RSLT;
    volatile uint32_t    INIT;
    volatile uint32_t    PLN;
    Volatile uint32_t    RXOR;
    Volatile uint32_t    IDATA;
    Volatile uint32_t    TEMP;
} CRC_SFRmap;
    
```

表 8-1 CRC 寄存器结构说明

寄存器	描述
CTL	CRC 控制寄存器，偏移:0x00
DATA	偏移:0x04
RSLT	偏移:0x08

INIT	偏移:0x0C
PLN	偏移:0x10
RXOR	偏移:0x14
IDATA;	偏移:0x18
TEMP	偏移:0x1C

CRC 配置信息结构体，定义于文件 kf32a_basic_crc.h 中，如下：

```
typedef struct
{
    uint32_t    m_CalUnitReset;
    uint32_t    m_InputSize;
    uint32_t    m_InputReverse;
    uint32_t    m_ResultReverse;
    uint32_t    m_Data;
    uint32_t    m_Result;
    uint32_t    m_InitialData;
    uint32_t    m_Polynomial;
    uint32_t    m_ResultXOR;
    uint32_t    m_IndepentData;
    uint32_t    m_Temp;
} CRC_InitTypeDef;
```

表 8-2 CRC 配置信息结构体说明

配置信息	描述
m_CalUnitReset	CRC 计算单元复位控制，取宏为“CRC 计算单元复位控制”中的一个。
m_InputSize	CRC 输入数据格式控制位，取宏为“CRC 输入数据格式控制位”中的一个。
m_InputReverse	CRC 输入数据反序控制位，取宏为“CRC 输入数据反序控制位”中的一个。
m_ResultReverse	CRC 结果反序控制，取宏为“CRC 结果反序控制”中的一个。
m_Data	CRC 输入数据值，取值 32 位数据，数据和需要右对齐。
m_Result	CRC 结果值，取值 32q 位数据，该值为只读。
m_InitialData	CRC 计算初始值，取值 32 位数据。
m_Polynomial	CRC 多项式值，取值 32 位，该值左对齐。
m_ResultXOR	CRC 结果异或 j 值，取值 32 位，该值左对齐。
m_IndepentData	CRC 独立数据寄存器值，取值 32 位。
m_Temp	CRC 缓存值，取值 32 位，该值为只读。

8.3 CRC 宏定义

CRC 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，CRC 其

他相关宏定义详见文件 kf32a_basic_crc.h 及函数参数描述。

8.4 CRC 库函数

表 8-3 CRC 固件库函数列表

序号	函数名	描述
1	CRC_Reset	复位 ADC 外设，使能外设时钟。
2	CRC_Configuration	模数转换模块(ADC)初始化配置。
3	CRC_Struct_Init	初始化 ADC 配置信息结构体。
4	CRC_INPUT_DATA	ADC0_DELAY 初始化配置。
5	CRC_GET_RESULT	初始化 ADC0 快慢交叉模式信息结构体。
6	CRC_SET_INITVALUE	配置 ADC 工作使能。
7	CRC_SET_PLN	ADC 模拟看门狗初始化配置。
8	CRC_SET_RXOR	初始化 ADC 模拟看门狗信息结构体。
9	CRC_SET_IDATA	配置扫描模式中模拟看门狗单一通道使能。
10	CRC_GET_TEMP	配置扫描模式使能。
11	CRC_SET_RSET	配置比较器的运放输入参考电压二分之一校准。

8.4.1 函数 CRC_Reset

表 8-4 函数 CRC_Reset

函数名	CRC_Reset
函数原型	void CRC_Reset ()
功能描述	CRC 外设复位，使能外设时钟。
输入参数 1	无
返回值	无
被调用函数	void RST_CTL3_Peripheral_Reset_Enable (uint32_t RST_CTL3_bit, FunctionalState NewState); void PCLK_CTL3_Peripheral_Clock_Enable (uint32_t PCLK_CTL3_bit, FunctionalState NewState);

8.4.2 函数 CRC_Configuration

表 8-5 函数 CRC_Configuration

函数名	CRC_Configuration
函数原型	void CRC_Configuration (CRC_InitTypeDef* CRCInitStruct)
功能描述	CRC 外设配置。
输入参数 1	CRCx: CRCInitStruct: CRC 配置信息

返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

8.4.3 函数 CRC_Struct_Init

表 8-6 函数 CRC_Struct_Init

函数名	CRC_Struct_Init
函数原型	void CRC_Struct_Init (CRC_InitTypeDef* CRCInitStruct)
功能描述	初始化 CRC 配置信息结构体。
输入参数 1	CRCInitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

8.4.4 函数 CRC_INPUT_DATA

表 8-7 函数 CRC_INPUT_DATA

函数名	CRC_INPUT_DATA
函数原型	void CRC_INPUT_DATA (uint32_t DATA)
功能描述	CRC 数据寄存器配置
输入参数 1	DATA: 数据值。
返回值	无
被调用函数	无

8.4.5 函数 CRC_GET_RESULT

表 8-8 函数 CRC_GET_RESULT

函数名	CRC_GET_RESULT
函数原型	uint32_t CRC_GET_RESULT ()
功能描述	读取 CRC 结果
输入参数 1	无
返回值	uint32_t 位宽的值
被调用函数	无

8.4.6 函数 CRC_SET_INITVALUE

表 8-9 函数 CRC_SET_INITVALUE

函数名	CRC_SET_INITVALUE
函数原型	void CRC_SET_INITVALUE (uint32_t DATA)

功能描述	设置 CRC 计算的初始值。
输入参数 1	uint32_t 位宽的值。
返回值	无
被调用函数	无

8.4.7 函数 CRC_SET_PLN

表 8-10 函数 CRC_SET_PLN

函数名	CRC_SET_PLN
函数原型	void CRC_SET_PLN (uint32_t DATA)
功能描述	设置 CRC 多项式。
输入参数 1	uint32_t 位宽的值。
返回值	无
被调用函数	无

8.4.8 函数 CRC_SET_RXOR

表 8-11 函数 CRC_SET_RXOR

函数名	CRC_SET_RXOR
函数原型	void CRC_SET_RXOR (uint32_t DATA)
功能描述	设置 CRC 结果异或值。
输入参数 1	uint32_t 位宽的值。
返回值	无
被调用函数	无

8.4.9 函数 CRC_SET_IDATA

表 8-12 函数 CRC_SET_IDATA

函数名	CRC_SET_IDATA
函数原型	void CRC_SET_IDATA (uint32_t DATA)
功能描述	设置 CRC 独立数据。
输入参数 1	uint32_t 位宽的值。
返回值	无
被调用函数	无

8.4.10 函数 CRC_GET_TEMP

表 8-13 函数 CRC_GET_TEMP

函数名	CRC_GET_TEMP
-----	--------------

函数原型	uint32_t CRC_GET_TEMP ()
功能描述	设置 CRC 缓存
输入参数 1	无
返回值	uint32_t 位宽的值
被调用函数	无

8.4.11 函数 CRC_SET_RSET

表 8-14 函数 CRC_SET_RSET

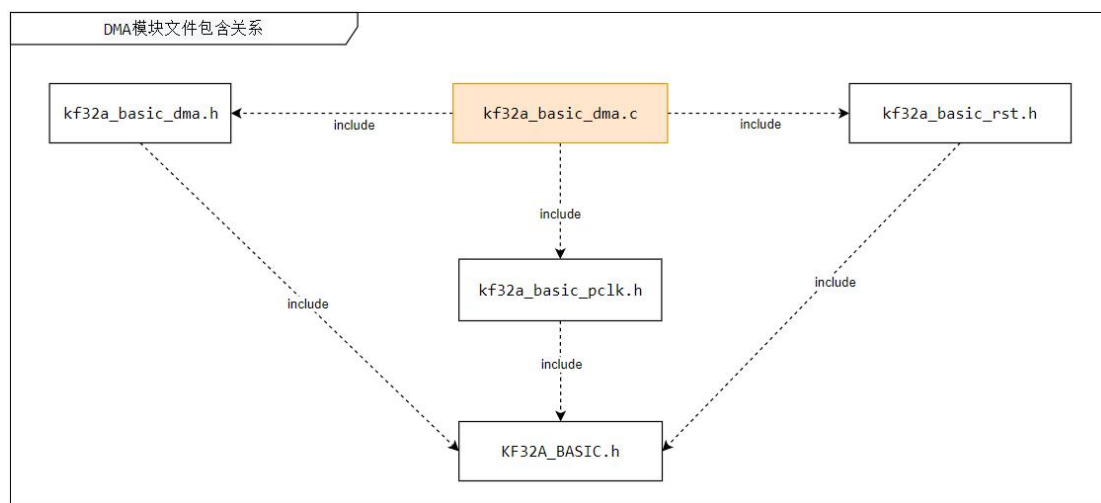
函数名	CRC_SET_RSET
函数原型	void CRC_SET_RSET ()
功能描述	CRC 计算单元复位。
输入参数 1	无
返回值	无
被调用函数	无

9 直接存储器（DMA）

KungFu32 内核提供了两个独立的直接存储器（Direct Memory Access）。每个 DMA 有 7 个通道。可用于 RAM 和 RAM 之间、RAM 和外设、外设和外设之间的数据传输。

9.1 文件引用关系

图 9-1 DMA 寄存器结构说明



9.2 DMA 寄存器结构

DMA 寄存器结构，DMA_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct DMA_MemMap {
    union {
        struct
        {
            volatile uint32_t    CTLR1;
            volatile uint32_t    CTLR2;
            volatile uint32_t    CTLR3;
            volatile uint32_t    CTLR4;
            volatile uint32_t    CTLR5;
            volatile uint32_t    CTLR6;
            volatile uint32_t    CTLR7;
        };
        volatile uint32_t    CTLR[7];
    };
    uint32_t RESERVED1;
    union {

```

```
struct
{
    volatile uint32_t    PADDR1;
    volatile uint32_t    PADDR2;
    volatile uint32_t    PADDR3;
    volatile uint32_t    PADDR4;
    volatile uint32_t    PADDR5;
    volatile uint32_t    PADDR6;
    volatile uint32_t    PADDR7;
};
volatile uint32_t    PADDR[7];
};
uint32_t    RESERVED2;
union {
    struct
    {
        volatile uint32_t    MADDR1;
        volatile uint32_t    MADDR2;
        volatile uint32_t    MADDR3;
        volatile uint32_t    MADDR4;
        volatile uint32_t    MADDR5;
        volatile uint32_t    MADDR6;
        volatile uint32_t    MADDR7;
    };
    volatile uint32_t    MADDR[7];
};
uint32_t    RESERVED3;
union {
    struct
    {
        volatile const uint32_t    CPAR1;
        volatile const uint32_t    CPAR2;
        volatile const uint32_t    CPAR3;
        volatile const uint32_t    CPAR4;
        volatile const uint32_t    CPAR5;
        volatile const uint32_t    CPAR6;
        volatile const uint32_t    CPAR7;
    };
    volatile const uint32_t    CPAR[7];
};
uint32_t    RESERVED4;
union {
```

```

struct
{
    volatile const uint32_t    CMAR1;
    volatile const uint32_t    CMAR2;
    volatile const uint32_t    CMAR3;
    volatile const uint32_t    CMAR4;
    volatile const uint32_t    CMAR5;
    volatile const uint32_t    CMAR6;
    volatile const uint32_t    CMAR7;
};
volatile const uint32_t    CMAR[7];
};
uint32_t    RESERVED5;
union {
    struct
    {
        volatile const uint32_t    NCT1;
        volatile const uint32_t    NCT2;
        volatile const uint32_t    NCT3;
        volatile const uint32_t    NCT4;
        volatile const uint32_t    NCT5;
        volatile const uint32_t    NCT6;
        volatile const uint32_t    NCT7;
    };
    volatile const uint32_t    NCT[7];
};
uint32_t    RESERVED6;
volatile uint32_t    LIFR;
volatile uint32_t    LIER;
}DMA_SFRmap;

```

表 9-1 DMA 寄存器结构说明

寄存器	描述
CTLR1	DMA 通道 1 控制寄存器,偏移:0x00
CTLR2	DMA 通道 2 控制寄存器,偏移:0x04
CTLR3	DMA 通道 3 控制寄存器,偏移:0x08
CTLR4	DMA 通道 4 控制寄存器,偏移:0x0C
CTLR5	DMA 通道 5 控制寄存器,偏移:0x10
CTLR6	DMA 通道 6 控制寄存器,偏移:0x14
CTLR7	DMA 通道 7 控制寄存器,偏移:0x18
RESERVED1	保留地址,偏移:0x1C
PADDR1	DMA 通道 1 外设地址寄存器,偏移:0x20
PADDR2	DMA 通道 2 外设地址寄存器,偏移:0x24

PADDR3	DMA 通道 3 外设地址寄存器,偏移:0x28
PADDR4	DMA 通道 4 外设地址寄存器,偏移:0x2C
PADDR5	DMA 通道 5 外设地址寄存器,偏移:0x30
PADDR6	DMA 通道 6 外设地址寄存器,偏移:0x34
PADDR7	DMA 通道 7 外设地址寄存器,偏移:0x38
RESERVED2	保留地址,偏移:0x3C
MADDR1	DMA 通道 1 存储器地址寄存器,偏移:0x40
MADDR2	DMA 通道 2 存储器地址寄存器,偏移:0x44
MADDR3	DMA 通道 3 存储器地址寄存器,偏移:0x48
MADDR4	DMA 通道 4 存储器地址寄存器,偏移:0x4C
MADDR5	DMA 通道 5 存储器地址寄存器,偏移:0x50
MADDR6	DMA 通道 6 存储器地址寄存器,偏移:0x54
MADDR7	DMA 通道 7 存储器地址寄存器,偏移:0x58
RESERVED3	保留地址,偏移:0x5C
CPAR1	DMA 通道 1 当前外设地址寄存器,偏移:0x60
CPAR2	DMA 通道 2 当前外设地址寄存器,偏移:0x64
CPAR3	DMA 通道 3 当前外设地址寄存器,偏移:0x68
CPAR4	DMA 通道 4 当前外设地址寄存器,偏移:0x6C
CPAR5	DMA 通道 5 当前外设地址寄存器,偏移:0x70
CPAR6	DMA 通道 6 当前外设地址寄存器,偏移:0x74
CPAR7	DMA 通道 7 当前外设地址寄存器,偏移:0x78
RESERVED4	保留地址,偏移:0x7C
CMAR1	DMA 通道 1 当前存储器地址寄存器,偏移:0x80
CMAR2	DMA 通道 2 当前存储器地址寄存器,偏移:0x84
CMAR3	DMA 通道 3 当前存储器地址寄存器,偏移:0x88
CMAR4	DMA 通道 4 当前存储器地址寄存器,偏移:0x8C
CMAR5	DMA 通道 5 当前存储器地址寄存器,偏移:0x90
CMAR6	DMA 通道 6 当前存储器地址寄存器,偏移:0x94
CMAR7	DMA 通道 7 当前存储器地址寄存器,偏移:0x98
RESERVED5	保留地址,偏移:0x9C
NCT1	DMA 通道 1 当前剩余数据寄存器,偏移:0xA0
NCT2	DMA 通道 2 当前剩余数据寄存器,偏移:0xA4
NCT3	DMA 通道 3 当前剩余数据寄存器,偏移:0xA8
NCT4	DMA 通道 4 当前剩余数据寄存器,偏移:0xAC
NCT5	DMA 通道 5 当前剩余数据寄存器,偏移:0xB0
NCT6	DMA 通道 6 当前剩余数据寄存器,偏移:0xB4
NCT7	DMA 通道 7 当前剩余数据寄存器,偏移:0xB8
RESERVED6	保留地址,偏移:0xBC
LIFR	DMA 中断标志寄存器,偏移:0xC0
LIER	DMA 中断使能寄存器,偏移:0xC4

DMA 配置信息结构体，DMA_InitTypeDef，定义于文件 kf32a_basic_dma.h 中，如下：

```
typedef struct
{
    uint8_t      m_Channel;
    uint8_t      m_Direction;
    uint8_t      m_PeripheralDataSize;
    uint8_t      m_MemoryDataSize;
    uint16_t     m_Priority;
    uint16_t     m_Number;
    FunctionalState m_PeripheralInc;
    FunctionalState m_MemoryInc;
    FunctionalState m_LoopMode;
    uint32_t     m_BlockMode;
    uint32_t     m_PeriphAddr;
    uint32_t     m_MemoryAddr;
}DMA_InitTypeDef;
```

表 9-2 DMA 配置信息结构体说明

配置信息	描述
m_Channel	DMA 通道选择，取值为宏“DMA 通道”中的一个
m_Direction	DMA 传输方向，取值为宏“传输方向”中的一个
m_PeripheralDataSize	外设数据位宽，取值为宏“传输数据位宽”中的一个
m_MemoryDataSize	存储器数据位宽，取值为宏“传输数据位宽”中的一个
m_Priority	DMA 通道优先级，取值为宏“DMA 通道优先级”中的一个
m_Number	传输数据个数，取值范围为 0~65535
m_PeripheralInc	外设地址增量模式使能，取值为 TRUE 或 FALSE
m_MemoryInc	存储器地址增量模式使能，取值为 TRUE 或 FALSE
m_LoopMode	循环模式使能，取值为 TRUE 或 FALSE
m_BlockMode	数据块传输模式，取值为宏“数据块传输模式”中的一个
m_PeriphAddr	外设起始地址，取值为 32 位数值
m_MemoryAddr	内存起始地址，取值为 32 位数值

9.3 DMA 宏定义

DMA 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，DMA 其他相关宏定义详见文件 kf32a_basic_dma.h 及函数参数描述。

9.4 DMA 库函数

表 9-3 DMA 固件库函数列表

序号	函数名	描述
1	DMA_Reset	复位 DMA 外设，使能外设时钟

2	DMA_Configuration	DMA 模块(DMA)初始化配置
3	DMA_Struct_Init	初始化 DMA 配置信息结构体
4	DMA_Transfer_Number_Config	配置 DMA 传输的数据量
5	DMA_Memory_To_Memory_Enable	配置存储器到存储器传输使能
6	DMA_Channel_Priority_Config	配置 DMA 通道优先级
7	DMA_Peripheral_Data_Width_Config	配置 DMA 外设数据位宽
8	DMA_Memory_Data_Width_Config	配置 DMA 存储器数据位宽
9	DMA_Peripheral_addr_increase_Enable	配置外设地址增量模式使能
10	DMA_Memory_addr_increase_Enable	配置存储器地址增量模式使能
11	DMA_Loop_Mode_Enable	配置循环模式使能
12	DMA_Transfer_Direction_Config	配置数据方向控制
13	DMA_Transfer_Mode_Config	配置数据块传输模式
14	DMA_Oneshot_Enable	配置 DMA 通道单次触发模式使能
15	DMA_Channel_Enable	配置 DMA 通道 Channel 使能
16	DMA_Peripheral_Start_Address_Config	配置 DMA 通道的起始外设地址
17	DMA_Memory_Start_Address_Config	配置 DMA 通道的起始存储器地址
18	DMA_Get_Peripheral_Current_Address	获取 DMA 通道的当前外设地址
19	DMA_Get_Memory_Current_Address	获取 DMA 通道的当前存储器地址
20	DMA_Get_Transfer_Number_Remain	获取 DMA 通道的当前剩余数据量
21	DMA_Get_INT_Flag	获取 DMA 通道传输中断标志
22	DMA_Clear_INT_Flag	清零 DMA 通道传输中断标志
23	DMA_Set_INT_Enable	配置 DMA 通道传输中断使能
24	DMA_Get_Error_Transfer_INT_Flag	获取 DMA 通道错误传输中断标志
25	DMA_Get_Half_Transfer_INT_Flag	获取 DMA 通道半传输中断标志
26	DMA_Get_Finish_Transfer_INT_Flag	获取 DMA 通道完成传输中断标志
27	DMA_Error_Transfer_INT_Enable	配置 DMA 通道错误传输中断使能
28	DMA_Half_Transfer_INT_Enable	配置 DMA 通道半传输中断使能
29	DMA_Finish_Transfer_INT_Enable	配置 DMA 通道完成传输中断使能

9.4.1 函数 DMA_Reset

表 9-4 函数 DMA_Reset

函数名	DMA_Reset
函数原型	void DMA_Reset (DMA_SFRmap* DMAx)
功能描述	复位 DMA 外设，使能外设时钟
输入参数 1	DMAx: 指向 DMA 内存结构的指针，取值为 DMA0_SFR 和 DMA1_SFR
返回值	无
被调用函数 1	void RST_CTL2_Peripheral_Reset_Enable (uint32_t RST_CTL2_bit, FunctionalState NewState)
被调用函数 2	void PCLK_CTL2_Peripheral_Clock_Enable (uint32_t PCLK_CTL2_bit,

	FunctionalState NewState)
--	---------------------------

9.4.2 函数 DMA_Configuration

表 9-5 函数 DMA_Configuration

函数名	DMA_Configuration
函数原型	void DMA_Configuration (DMA_SFRmap* DMAx,dDMA_InitTypeDef* dmaInitStruct)
功能描述	DMA 模块(DMA)初始化配置
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	dmaInitStruct: DMA 模块配置信息结构体指针
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

9.4.3 函数 DMA_Struct_Init

表 9-6 函数 DMA_Struct_Init

函数名	DMA_Struct_Init
函数原型	void DMA_Struct_Init (DMA_InitTypeDef* dmaInitStruct)
功能描述	初始化 DMA 配置信息结构体
输入参数 1	dmaInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

9.4.4 函数 DMA_Transfer_Number_Config

表 9-7 函数 DMA_Transfer_Number_Config

函数名	DMA_Transfer_Number_Config
函数原型	void DMA_Transfer_Number_Config (DMA_SFRmap* DMAx, uint32_t Channel, uint16_t Number)
功能描述	配置 DMA 传输的数据量
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6

	DMA_CHANNEL_7: 通道 7
输入参数 3	Number: 需要传输的数据个数, 取值范围为 0~65535
返回值	无
被调用函数	无

9.4.5 函数 DMA_Memory_To_Memory_Enable

表 9-8 函数 DMA_Memory_To_Memory_Enable

函数名	DMA_Memory_To_Memory_Enable
函数原型	void DMA_Memory_To_Memory_Enable (DMA_SFRmap* DMAx,uint32_t Channel, FunctionalState NewState)
功能描述	配置存储器到存储器传输使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: 存储器到存储器传输使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.6 函数 DMA_Channel_Priority_Config

表 9-9 函数 DMA_Channel_Priority_Config

函数名	DMA_Channel_Priority_Config
函数原型	void DMA_Channel_Priority_Config (DMA_SFRmap* DMAx,uint32_t Channel, uint32_t Priority)
功能描述	配置 DMA 通道优先级
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7

输入参数 3	Priority: 通道优先级, 取值为: DMA_CHANNEL_LOWER: 低优先级 DMA_CHANNEL_MEDIUM: 中优先级 DMA_CHANNEL_HIGHER: 高优先级 DMA_CHANNEL_TOP: 最高优先级
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

9.4.7 函数 DMA_Peripheral_Data_Width_Config

表 9-10 函数 DMA_Peripheral_Data_Width_Config

函数名	DMA_Peripheral_Data_Width_Config
函数原型	void DMA_Peripheral_Data_Width_Config (DMA_SFRmap* DMAx,uint32_t Channel, uint32_t Width)
功能描述	配置 DMA 外设数据位宽
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	Width: 外设数据位宽, 取值为: DMA_DATA_WIDTH_8_BITS: 数据为 8 位宽 DMA_DATA_WIDTH_16_BITS: 数据为 16 位宽 DMA_DATA_WIDTH_32_BITS: 数据为 32 位宽
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

9.4.8 函数 DMA_Memory_Data_Width_Config

表 9-11 函数 DMA_Memory_Data_Width_Config

函数名	DMA_Memory_Data_Width_Config
函数原型	void DMA_Memory_Data_Width_Config (DMA_SFRmap* DMAx,uint32_t Channel, uint32_t Width)
功能描述	配置 DMA 存储器数据位宽
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR

输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	Width: 存储器数据位宽, 取值为: DMA_DATA_WIDTH_8_BITS: 数据为 8 位宽 DMA_DATA_WIDTH_16_BITS: 数据为 16 位宽 DMA_DATA_WIDTH_32_BITS: 数据为 32 位宽
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

9.4.9 函数 DMA_Peripheral_addr_increase_Enable

表 9-12 函数 DMA_Peripheral_addr_increase_Enable

函数名	DMA_Peripheral_addr_increase_Enable
函数原型	void DMA_Peripheral_addr_increase_Enable (DMA_SFRmap* DMAx, uint32_t Channel, FunctionalState NewState)
功能描述	DMA 模块(DMA)初始化配置
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: 外设地址增量模式使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.10 函数 DMA_Memory_addr_increase_Enable

表 9-13 函数 DMA_Memory_addr_increase_Enable

函数名	DMA_Memory_addr_increase_Enable
函数原型	void DMA_Memory_addr_increase_Enable (DMA_SFRmap* DMAx, uint32_t

	Channel, FunctionalState NewState)
功能描述	配置存储器地址增量模式使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: 外设地址增量模式使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.11 函数 DMA_Loop_Mode_Enable

表 9-14 函数 DMA_Loop_Mode_Enable

函数名	DMA_Loop_Mode_Enable
函数原型	void DMA_Loop_Mode_Enable (DMA_SFRmap* DMAx,uint32_t Channel, FunctionalState NewState)
功能描述	配置循环模式使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: 循环模式使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

9.4.12 函数 DMA_Transfer_Direction_Config

表 9-15 函数 DMA_Transfer_Direction_Config

函数名	DMA_Transfer_Direction_Config
函数原型	void DMA_Transfer_Direction_Config (DMA_SFRmap* DMAx,uint32_t Channel, uint32_t Direction)

功能描述	配置数据方向控制
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	Direction: 数据方向控制状态, 取值为: DMA_PERIPHERAL_TO_MEMORY: 数据从外设地址读取, 存入存储器地址 DMA_MEMORY_TO_PERIPHERAL: 数据从存储器地址读取, 存入外设地址
返回值	无
被调用函数	无

9.4.13 函数 DMA_Transfer_Mode_Config

表 9-16 函数 DMA_Transfer_Mode_Config

函数名	DMA_Transfer_Mode_Config
函数原型	void DMA_Transfer_Mode_Config (DMA_SFRmap* DMAx, uint32_t Channel, uint32_t BlockMode)
功能描述	配置数据块传输模式
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	BlockMode: 数据块传输模式, 取值为: DMA_TRANSFER_BYTE: 数据按照字节传输 DMA_TRANSFER_BLOCK: 数据按照块传输
返回值	无
被调用函数	无

9.4.14 函数 DMA_OneShot_Enable

表 9-17 函数 DMA_Transfer_Mode_Config

函数名	DMA_OneShot_Enable
函数原型	void DMA_OneShot_Enable (DMA_SFRmap* DMAx,uint32_t Channel, FunctionalState NewState)
功能描述	配置 DMA 通道单次触发模式使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: DMA 通道 Channel 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.15 函数 DMA_Channel_Enable

表 9-18 函数 DMA_Channel_Enable

函数名	DMA_Channel_Enable
函数原型	void DMA_Channel_Enable (DMA_SFRmap* DMAx,uint32_t Channel, FunctionalState NewState)
功能描述	配置数据块传输模式
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: DMA 通道 Channel 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.16 函数 DMA_Peripheral_Start_Address_Config

表 9-19 函数 DMA_Peripheral_Start_Address_Config

函数名	DMA_Peripheral_Start_Address_Config
函数原型	void DMA_Peripheral_Start_Address_Config (DMA_SFRmap* DMAx, uint32_t Channel, uint32_t Address)
功能描述	配置 DMA 通道的起始外设地址
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	Address: DMA 通道的起始外设地址, 取值为 32 位有效数值
返回值	无
被调用函数	无

9.4.17 函数 DMA_Memory_Start_Address_Config

表 9-20 函数 DMA_Transfer_Mode_Config

函数名	DMA_Memory_Start_Address_Config
函数原型	void DMA_Memory_Start_Address_Config (DMA_SFRmap* DMAx, uint32_t Channel, uint32_t Address)
功能描述	配置 DMA 通道的起始存储器地址
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	Address: DMA 通道的起始存储器地址, 取值为 32 位有效数值。
返回值	无
被调用函数	无

9.4.18 函数 DMA_Get_Peripheral_Current_Address

表 9-21 函数 DMA_Get_Peripheral_Current_Address

函数名	DMA_Get_Peripheral_Current_Address
函数原型	uint32_t DMA_Get_Peripheral_Current_Address (DMA_SFRmap* DMAx, uint32_t Channel)
功能描述	获取 DMA 通道的当前外设地址
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
返回值	DMA 通道的当前外设地址, 取值为 32 位有效数值。
被调用函数	无

9.4.19 函数 DMA_Get_Memory_Current_Address

表 9-22 函数 DMA_Get_Memory_Current_Address

函数名	DMA_Get_Memory_Current_Address
函数原型	uint32_t DMA_Get_Memory_Current_Address (DMA_SFRmap* DMAx, uint32_t Channel)
功能描述	获取 DMA 通道的当前存储器地址
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
返回值	DMA 通道的当前存储器地址, 取值为 32 位有效数值
被调用函数	无

9.4.20 函数 DMA_Get_Transfer_Number_Remain

表 9-23 函数 DMA_Get_Transfer_Number_Remain

函数名	DMA_Get_Transfer_Number_Remain
函数原型	uint16_t DMA_Get_Transfer_Number_Remain (DMA_SFRmap* DMAx, uint32_t Channel)
功能描述	获取 DMA 通道的当前剩余数据量
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
返回值	DMA 通道的当前剩余数据量, 取值为 16 位有效数值。
被调用函数	无

9.4.21 函数 DMA_Get_INT_Flag

表 9-24 函数 DMA_Get_INT_Flag

函数名	DMA_Get_INT_Flag
函数原型	FlagStatus DMA_Get_INT_Flag (DMA_SFRmap* DMAx, uint32_t Channel, uint32_t InterruptType)
功能描述	获取 DMA 通道传输中断标志
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	InterruptType: DMA 通道中断类型, 取值范围为: DMA_INT_FINISH_TRANSFER: 完成传输中断 DMA_INT_HALF_TRANSFER: 半传输中断 DMA_INT_ERROR_TRANSFER: 错误传输中断
返回值	1:传输发生错误, 0:传输无错误
被调用函数	无

9.4.22 函数 DMA_Clear_INT_Flag

表 9-25 函数 DMA_Clear_INT_Flag

函数名	DMA_Clear_INT_Flag
函数原型	void DMA_Clear_INT_Flag (DMA_SFRmap* DMAx, uint32_t Channel, uint32_t InterruptType)
功能描述	清零 DMA 通道传输中断标志
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	InterruptType: DMA 通道中断类型, 取值范围为下面提供的一个或多个组合: DMA_INT_FINISH_TRANSFER: 完成传输中断 DMA_INT_HALF_TRANSFER: 半传输中断 DMA_INT_ERROR_TRANSFER: 错误传输中断
返回值	无
被调用函数	无

9.4.23 函数 DMA_Set_INT_Enable

表 9-26 函数 DMA_Set_INT_Enable

函数名	DMA_Set_INT_Enable
函数原型	void DMA_Set_INT_Enable (DMA_SFRmap* DMAx, uint32_t Channel, uint32_t InterruptType, FunctionalState NewState)
功能描述	配置 DMA 通道传输中断使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6

	DMA_CHANNEL_7: 通道 7
输入参数 3	InterruptType: DMA 通道中断类型, 取值范围为下面提供的一个或多个组合: DMA_INT_FINISH_TRANSFER: 完成传输中断 DMA_INT_HALF_TRANSFER: 半传输中断 DMA_INT_ERROR_TRANSFER: 错误传输中断
输入参数 4	NewState: DMA 通道错误传输中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.24 函数 DMA_Get_Error_Transfer_INT_Flag

表 9-27 函数 DMA_Get_Error_Transfer_INT_Flag

函数名	DMA_Get_Error_Transfer_INT_Flag
函数原型	FlagStatus DMA_Get_Error_Transfer_INT_Flag (DMA_SFRmap* DMAx, uint32_t Channel)
功能描述	获取 DMA 通道错误传输中断标志
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
返回值	1:传输发生错误, 0:传输无错误。
被调用函数	无

9.4.25 函数 DMA_Get_Half_Transfer_INT_Flag

表 9-28 函数 DMA_Get_Half_Transfer_INT_Flag

函数名	DMA_Get_Half_Transfer_INT_Flag
函数原型	FlagStatus DMA_Get_Half_Transfer_INT_Flag (DMA_SFRmap* DMAx, uint32_t Channel)
功能描述	获取 DMA 通道半传输中断标志
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3

	DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
返回值	1:传输已完成一半, 0:传输未完成一半。
被调用函数	无

9.4.26 函数 DMA_Get_Finish_Transfer_INT_Flag

表 9-29 函数 DMA_Get_Finish_Transfer_INT_Flag

函数名	DMA_Get_Finish_Transfer_INT_Flag
函数原型	FlagStatus DMA_Get_Finish_Transfer_INT_Flag (DMA_SFRmap* DMAx, uint32_t Channel)
功能描述	获取 DMA 通道完成传输中断标志
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
返回值	1:传输已完成, 0:传输未完成
被调用函数	无

9.4.27 函数 DMA_Error_Transfer_INT_Enable

表 9-30 函数 DMA_Error_Transfer_INT_Enable

函数名	DMA_Error_Transfer_INT_Enable
函数原型	void DMA_Error_Transfer_INT_Enable (DMA_SFRmap* DMAx, uint32_t Channel, FunctionalState NewState)
功能描述	配置 DMA 通道错误传输中断使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6

	DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: DMA 通道错误传输中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.28 函数 DMA_Half_Transfer_INT_Enable

表 9-31 函数 DMA_Half_Transfer_INT_Enable

函数名	DMA_Half_Transfer_INT_Enable
函数原型	void DMA_Half_Transfer_INT_Enable (DMA_SFRmap* DMAx, uint32_t Channel, FunctionalState NewState)
功能描述	配置 DMA 通道半传输中断使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7
输入参数 3	NewState: DMA 通道半传输中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

9.4.29 函数 DMA_Finish_Transfer_INT_Enable

表 9-32 函数 DMA_Finish_Transfer_INT_Enable

函数名	DMA_Finish_Transfer_INT_Enable
函数原型	void DMA_Finish_Transfer_INT_Enable (DMA_SFRmap* DMAx, uint32_t Channel, FunctionalState NewState)
功能描述	配置 DMA 通道完成传输中断使能
输入参数 1	DMAx: 指向 DMA 内存结构的指针, 取值为 DMA0_SFR 和 DMA1_SFR
输入参数 2	Channel: DMA 通道选择, 取值范围为: DMA_CHANNEL_1: 通道 1 DMA_CHANNEL_2: 通道 2 DMA_CHANNEL_3: 通道 3 DMA_CHANNEL_4: 通道 4 DMA_CHANNEL_5: 通道 5 DMA_CHANNEL_6: 通道 6 DMA_CHANNEL_7: 通道 7

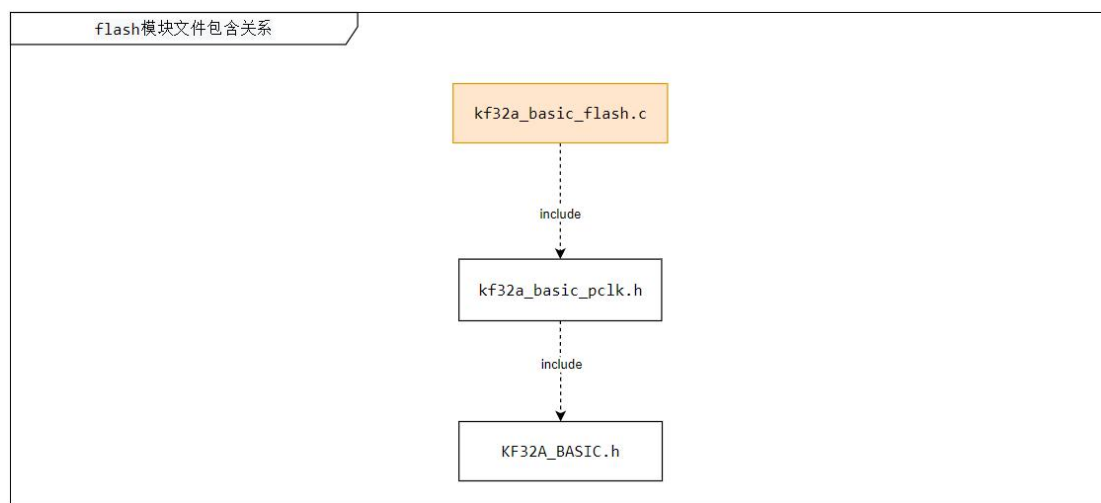
输入参数 3	NewState: DMA 通道完成传输中断使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

10 闪存（FLASH）

KungFu32 提供最大 512 Kbyte 的空间大小，以 64bit 为单元进行读写访问，支持指令预取，纠 1 位错的 ECC。

10.1 文件引用关系

图 10-1 FLASH 寄存器结构说明



10.2 FLASH 寄存器结构

FLASH 寄存器结构，FLASH_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct FLASH_MemMap {
    volatile uint32_t    ISPCON0;
    volatile uint32_t    ISPCON1;
    volatile uint32_t    ISPCMD;
    volatile uint32_t    ISPTRG;
    uint32_t    RESERVED1;
    volatile uint32_t    CFG;
    uint32_t    RESERVED2;
    volatile uint32_t    ISPADDR;
    volatile uint32_t    STATE;
    uint32_t    RESERVED3;
    volatile uint32_t    NVMUNLOCK;
    volatile uint32_t    PROUNLOCK;
    volatile uint32_t    CFGUNLOCK;
    uint32_t    RESERVED4;
    volatile uint32_t    CSSTART;

```

```
volatile uint32_t    CSSTOP;
volatile uint32_t    CSRES[4];
}FLASH_SFRmap;
```

表 10-1 FLASH 寄存器结构说明

寄存器	描述
ISPCON0	FLASH 控制寄存器,偏移:0x00
ISPCON1	FLASH 控制寄存器,偏移:0x04
ISPCMD	FLASH 控制寄存器,偏移:0x08
ISPTRG	FLASH 控制寄存器,偏移:0x0C
RESERVED1	保留地址,偏移:0x10
CFG	FLASH 控制寄存器,偏移:0x14
RESERVED2	保留地址,偏移:0x18
ISPADDR	FLASH 地址寄存器,偏移:0x1C
STATE	FLASH 状态寄存器,偏移:0x20
RESERVED3	保留地址,偏移:0x24
NVMUNLOCK	FLASH 解锁 KEY 寄存器,偏移:0x28
PROUNLOCK	FLASH 解锁 KEY 寄存器,偏移:0x2C
CFGUNLOCK	FLASH 解锁 KEY 寄存器,偏移:0x30
RESERVED4	保留地址,偏移:0x34
CSSTART	FLASHChecksum 首地址寄存器,偏移:0x38
CSSTOP	FLASHChecksum 尾地址寄存器,偏移:0x3C
CSRES[4]	FLASHChecksum 结果寄存器,偏移:0x40

FLASH 锁定解锁状态, LockStatus, 定义于文件 kf32a_basic_flash.h 中, 如下:

```
typedef enum
{
    LOCK = 0,
    UNLOCK = !LOCK
} LockStatus;
```

表 10-2 FLASH 锁定解锁状态

状态	描述
LOCK	0
UNLOCK	1

FLASH CheckSum 结果数据结构体, FLASH_CheckSumResult, 定义于文件 kf32a_basic_flash.h 中, 如下:

```
typedef union CheckSumStruct
{
    uint32_t    m_ResultWord[4];
    uint32_t    m_ResultShort[8];
    uint32_t    m_ResultByte[16];
}
```

```
}FLASH_CheckSumResult;
```

表 10-3 FLASH CheckSum 结果数据结构体说明

配置信息	描述
m_ResultWord[4]	Checksum 的 128 位结果，字访问。
m_ResultShort[8]	Checksum 的 128 位结果，半字访问。
m_ResultByte[16]	Checksum 的 128 位结果，字节访问

FLASH 编程信息结构体，FLASH_ProgramTypeDef，定义于文件 kf32a_basic_flash.h 中，如下：

```
typedef struct
{
    uint32_t    m_Mode;
    uint32_t    m_Zone;
    uint32_t    m_Addr;
    uint32_t    m_WriteSize;
    uint32_t *   m_Data;
}FLASH_ProgramTypeDef;
```

表 10-4 FLASH 编程信息结构体说明

配置信息	描述
m_Mode	编程模式，取值宏“FLASH 编程模式”中的一个。
m_Zone	编程区域，取值宏“FLASH 编程区域”中的一个。
m_Addr	编程地址，取值为 0x0~0xFFFFF，硬件忽略低 2 位。
m_WriteSize	编程长度，单位：双字(64 位)取值为 1~63。
m_Data	编程数据指针，指向待写数据。

10.3 FLASH 宏定义

FLASH 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, FLASH 其他相关宏定义详见文件 kf32a_basic_flash.h 及函数参数描述。

10.4 FLASH 库函数

表 10-5 FLASH 固件库函数列表

序号	函数名	描述
1	CHECK_RESTRICTION_RAM	用户检查断点功能
2	SFR_Config_RAM	写特殊功能寄存器
3	FLASH_Get_NonVolatile_Memory_Unlock_Stat us_RAM	读 FLASH 编程组件解锁状态。

4	FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM	清零 FLASH 编程组件解锁状态位。
5	FLASH_Clear_NonVolatile_Memory_Unlock_Status	清零 FLASH 编程组件解锁状态位。
6	FLASH_Unlock_ISP_RAM	置位 FLASH 编程组件解锁状态位，解锁 ISP 寄存器。
7	FLASH_Unlock_ISP	置位 FLASH 编程组件解锁状态位，解锁 ISP 寄存器。
8	FLASH_Get_Flash_Unlock_Status_RAM	读 FLASH 解锁状态。
9	FLASH_Clear_Flash_Unlock_Status_RAM	清零 FLASH 解锁状态位。
10	FLASH_Clear_Flash_Unlock_Status	清零 FLASH 解锁状态位。
11	FLASH_Unlock_Code_RAM	置位 FLASH 解锁状态位，解锁 FLASH 程序区。
12	FLASH_Unlock_Code	置位 FLASH 解锁状态位，解锁 FLASH 程序区。
13	FLASH_Get_Config_Unlock_Status_RAM	读 FLASH 配置区解锁状态。
14	FLASH_Clear_Config_Unlock_Status_RAM	清零 FLASH 配置区解锁状态位。
15	FLASH_Clear_Config_Unlock_Status	清零 FLASH 配置区解锁状态位。
16	FLASH_Unlock_User_Config_RAM	置位 FLASH 配置区解锁状态位，用户配置区。
17	FLASH_Unlock_User_Config	置位 FLASH 配置区解锁状态位，用户配置区。
18	FLASH_Data_Write_Enable_RAM	NewState: FLASH 写缓存使能状态，取值为 TRUE 或 FALSE。
19	FLASH_Zone_Config_RAM	配置 FLASH 信息区选择。
20	FLASH_Zone_Config	配置 FLASH 信息区选择。
21	FLASH_Standby_Mode_Config_RAM	配置 FLASH 模式。
22	FLASH_Read_Mode_Config_RAM	配置 FLASH 读模式。
23	FLASH_Calibration_Update_Enable_RAM	配置 FLASH 校准信息更新使能。
24	FLASH_Information_Zone_Wipe_Unlock_Config_RAM	配置 FLASH 芯片信息区擦写解锁状态。
25	FLASH_Half_Page_Write_Size_Config_RAM	配置 FLASH 半页编程模式下需要写入的字数。
26	FLASH_Program_Cmd_Config_RAM	配置 FLASH 命令选择。
27	FLASH_Executive_Cmd_RAM	执行当前 FLASH_ISPCMD 寄存器设定的命令，即 FLASH_Program_Cmd_Config 函数配置的命令。
28	FLASH_Executive_Cmd	执行当前 FLASH_ISPCMD 寄存器设定的命令，即 FLASH_Program_Cmd_Config 函数

		配置的命令
29	FLASH_NonVolatile_Memory_ECC_Enable_RAM	配置 FLASH 非易失性存储器 ECC 使能。
30	FLASH_Linear_Prefetch_Enable_RAM	配置 FLASH 线性预取使能。
31	FLASH_Period_Number_Config_RAM	配置 FLASH 访问周期。
32	FLASH_Program_Addr_Config_RAM	配置 FLASH 编程和行擦期间的地址
33	FLASH_Get_Program_Status_RAM	读 FLASH 编程状态。
34	FLASH_Get_Program_Status	读 FLASH 编程状态。
35	FLASH_Get_Wipe_Status_RAM	读 FLASH 擦写状态
36	FLASH_Get_Wipe_Status	读 FLASH 擦写状态。
37	FLASH_Get_Compute_Complete_Status_RAM	读 FLASH 计算完成状态。
38	FLASH_Clear_Compute_Complete_Status_RAM	清零 FLASH 计算完成状态。
39	FLASH_Get_CFG_Error_Flag_RAM	读 FLASH CFG 编程错误标志
40	FLASH_Clear_CFG_Error_Flag_RAM	清零 FLASH CFG 编程错误标志。
41	FLASH_CheckSum_Addr_Config_RAM	配置 FLASH CheckSum 首尾地址
42	FLASH_Start_SIG_Compute_Enable_RAM	配置 FLASH 启动 SIG 计算使能。
43	FLASH_Get_CheckSum_Result_RAM	获取 FLASH CheckSum 的结果。
44	FLASH_Wipe_Configuration_RAM	FLASH 擦除操作配置，用户需要确认 FLASH 擦写空闲状态。
45	FLASH_Wipe_Configuration	FLASH 擦除操作配置，用户需要确认 FLASH 擦写空闲状态。
46	FLASH_Program_Configuration_RAM	FLASH 编程操作配置，用户需要确认 FLASH 擦写空闲状态
47	FLASH_Program_Configuration	FLASH 编程操作配置，用户需要确认 FLASH 擦写空闲状态。
48	Read_Flash_or_CFR_RAM	读取 flash 的程序区或者信息区 (RAM)
49	Read_Flash_or_CFR	读取 flash 的程序区或者信息区
50	Read_Soft_Device_ID1	读取芯片唯一 ID
51	Read_Soft_Device_ID2	读取芯片唯一 ID
52	Read_Soft_Device_ID3	读取芯片唯一 ID
53	Read_Soft_Device_ID4	读取芯片唯一 ID

10.4.1 函数 CHECK_RESTRICTION_RAM

表 10-6 函数 CHECK_RESTRICTION_RAM

函数名	CHECK_RESTRICTION_RAM
函数原型	void __attribute__((section(".indata"))) CHECK_RESTRICTION_RAM(int expr)

功能描述	用户检查断点功能
输入参数 1	expr:输入 0 则错误
返回值	无
被调用函数 1	无

10.4.2 函数 SFR_Config_RAM

表 10-7 函数 SFR_Config_RAM

函数名	SFR_Config_RAM
函数原型	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)
功能描述	写特殊功能寄存器 RAM
输入参数 1	SfrMem:: 寄存器当前值
输入参数 2	SfrMask: 操作掩码
输入参数 3	WriteVal: 需改动的值
返回值	寄存器新的值
被调用函数 1	无

10.4.3 函数 FLASH_Get_NonVolatile_Memory_Unlock_Status_RAM

表 10-8 函数 FLASH_Get_NonVolatile_Memory_Unlock_Status_RAM

函数名	FLASH_Get_NonVolatile_Memory_Unlock_Status_RAM
函数原型	LockStatus __attribute__((section(".indata"))) FLASH_Get_NonVolatile_Memory_Unlock_Status_RAM (void)
功能描述	读 FLASH 编程组件解锁状态。
输入参数 1	无
返回值	锁定状态，1：解锁状态，0：锁定状态。
被调用函数 1	无

10.4.4 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM

表 10-9 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM

函数名	FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM (void)
功能描述	清零 FLASH 编程组件解锁状态位。
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.5 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status

表 10-10 函数 FLASH_Clear_NonVolatile_Memory_Unlock_Status

函数名	FLASH_Clear_NonVolatile_Memory_Unlock_Status
函数原型	void FLASH_Clear_NonVolatile_Memory_Unlock_Status (void)
功能描述	清零 FLASH 编程组件解锁状态位。
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.6 函数 FLASH_Unlock_ISP_RAM

表 10-11 函数 FLASH_Unlock_ISP_RAM

函数名	FLASH_Unlock_ISP_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Unlock_ISP_RAM (void)
功能描述	置位 FLASH 编程组件解锁状态位，解锁 ISP 寄存器。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.7 函数 FLASH_Unlock_ISP

表 10-12 函数 FLASH_Unlock_ISP

函数名	FLASH_Unlock_ISP
函数原型	void FLASH_Unlock_ISP (void)
功能描述	置位 FLASH 编程组件解锁状态位，解锁 ISP 寄存器。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.8 函数 FLASH_Get_Flash_Unlock_Status_RAM

表 10-13 函数 FLASH_Get_Flash_Unlock_Status_RAM

函数名	FLASH_Get_Flash_Unlock_Status_RAM
函数原型	LockStatus __attribute__((section(".indata"))) FLASH_Get_Flash_Unlock_Status_RAM (void)

功能描述	读 FLASH 解锁状态。
输入参数 1	无
返回值	锁定状态，1：解锁状态，0：锁定状态。
被调用函数 1	无

10.4.9 函数 FLASH_Clear_Flash_Unlock_Status_RAM

表 10-14 函数 FLASH_Clear_Flash_Unlock_Status_RAM

函数名	FLASH_Clear_Flash_Unlock_Status_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Clear_Flash_Unlock_Status_RAM (void)
功能描述	清零 FLASH 解锁状态位。
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.10 函数 FLASH_Clear_Flash_Unlock_Status

表 10-15 函数 FLASH_Clear_Flash_Unlock_Status

函数名	FLASH_Clear_Flash_Unlock_Status
函数原型	void FLASH_Clear_Flash_Unlock_Status (void)
功能描述	清零 FLASH 解锁状态位。
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.11 函数 FLASH_Unlock_Code_RAM

表 10-16 函数 FLASH_Unlock_Code_RAM

函数名	FLASH_Unlock_Code_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Unlock_Code_RAM (void)
功能描述	置位 FLASH 解锁状态位，解锁 FLASH 程序区。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.12 函数 FLASH_Unlock_Code

表 10-17 函数 FLASH_Unlock_Code

函数名	FLASH_Unlock_Code
函数原型	void FLASH_Unlock_Code (void)
功能描述	置位 FLASH 解锁状态位，解锁 FLASH 程序区。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.13 函数 FLASH_Get_Config_Unlock_Status_RAM

表 10-18 函数 FLASH_Get_Config_Unlock_Status_RAM

函数名	FLASH_Get_Config_Unlock_Status_RAM
函数原型	LockStatus __attribute__((section(".indata"))) FLASH_Get_Config_Unlock_Status_RAM (void)
功能描述	读 FLASH 配置区解锁状态。
输入参数 1	无
返回值	锁定状态，1：解锁状态，0：锁定状态。
被调用函数 1	无

10.4.14 函数 FLASH_Clear_Config_Unlock_Status_RAM

表 10-19 函数 FLASH_Clear_Config_Unlock_Status_RAM

函数名	FLASH_Clear_Config_Unlock_Status_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Clear_Config_Unlock_Status_RAM (void)
功能描述	清零 FLASH 配置区解锁状态位。
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.15 函数 FLASH_Clear_Config_Unlock_Status

表 10-20 函数 FLASH_Clear_Config_Unlock_Status

函数名	FLASH_Clear_Config_Unlock_Status
函数原型	void FLASH_Clear_Config_Unlock_Status (void)
功能描述	清零 FLASH 配置区解锁状态位。
输入参数 1	无

返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.16 函数 FLASH_Unlock_User_Config_RAM

表 10-21 函数 FLASH_Unlock_User_Config_RAM

函数名	FLASH_Unlock_User_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Unlock_User_Config_RAM (void)
功能描述	置位 FLASH 配置区解锁状态位，用户配置区。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.17 函数 FLASH_Unlock_User_Config

表 10-22 函数 FLASH_Unlock_User_Config

函数名	FLASH_Unlock_User_Config
函数原型	void FLASH_Unlock_User_Config(void)
功能描述	置位 FLASH 配置区解锁状态位，用户配置区。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.18 函数 FLASH_Data_Write_Enable_RAM

表 10-23 函数 FLASH_Data_Write_Enable_RAM

函数名	FLASH_Data_Write_Enable_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Data_Write_Enable_RAM (FunctionalState NewState)
功能描述	配置 FLASH 写缓存使能。
输入参数 1	NewState: FLASH 写缓存使能状态，取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

10.4.19 函数 FLASH_Zone_Config_RAM

表 10-24 函数 FLASH_Zone_Config_RAM

函数名	FLASH_Zone_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Zone_Config_RAM (uint32_t ZoneSelect)
功能描述	配置 FLASH 信息区选择。
输入参数 1	输入 ZoneSelect: FLASH 信息区选择, 取值为: FLASH_NVM_CODE_ZONE: FLASH 程序区 FLASH_NVM_INFORMATION_ZONE: FLASH_NVM_CODE_ZONE
返回值	无
被调用函数 1	无

10.4.20 函数 FLASH_Zone_Config

表 10-25 函数 FLASH_Zone_Config

函数名	FLASH_Zone_Config
函数原型	void FLASH_Zone_Config(uint32_t ZoneSelect)
功能描述	配置 FLASH 信息区选择。
输入参数 1	输入 ZoneSelect: FLASH 信息区选择, 取值为: FLASH_NVM_CODE_ZONE: FLASH 程序区 FLASH_NVM_INFORMATION_ZONE: FLASH_NVM_CODE_ZONE
返回值	无
被调用函数 1	无

10.4.21 函数 FLASH_Standby_Mode_Config_RAM

表 10-26 函数 FLASH_Standby_Mode_Config_RAM

函数名	FLASH_Standby_Mode_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Standby_Mode_Config_RAM (uint32_t ModeSelect)
功能描述	配置 FLASH 模式。
输入参数 1	输入 ModeSelect: FLASH 信息区选择, 取值为: FLASH_MODE_NORMAL: 均为正常模式 FLASH_MODE_STANDBY1: STANDBY1 进入 Standby 模式 FLASH_MODE_STANDBY2: STANDBY2 进入正常模式 FLASH_MODE_STANDBY1_STANDBY2: 均进入 Standby 模式
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.22 函数 FLASH_Read_Mode_Config_RAM

表 10-27 函数 FLASH_Read_Mode_Config_RAM

函数名	FLASH_Read_Mode_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Read_Mode_Config_RAM (uint32_t ReadMode)
功能描述	配置 FLASH 读模式。
输入参数 1	ReadMode: FLASH 读模式选择, 取值为: FLASH_READ_MODE_NORMAL: 正常模式 FLASH_READ_MODE_RECALL: RECALL 模式
返回值	无
被调用函数 1	无

10.4.23 函数 FLASH_Calibration_Updata_Enable_RAM

表 10-28 函数 FLASH_Calibration_Updata_Enable_RAM

函数名	FLASH_Calibration_Updata_Enable_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Calibration_Updata_Enable_RAM (FunctionalState NewState)
功能描述	配置 FLASH 校准信息更新使能。
输入参数 1	NewState: FLASH 校准信息更新使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.24 函数 FLASH_Information_Zone_Wipe_Unlock_Config_RAM

表 10-29 函数 FLASH_Information_Zone_Wipe_Unlock_Config_RAM

函数名	FLASH_Information_Zone_Wipe_Unlock_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Information_Zone_Wipe_Unlock_Config_RAM (LockStatus NewState)
功能描述	配置 FLASH 芯片信息区擦写解锁状态。
输入参数 1	输入 NewState: FLASH 芯片信息区擦写解锁状态, 取值为: LOCK: 上锁状态 UNLOCK: 解锁状态
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.25 函数 FLASH_Half_Page_Write_Size_Config_RAM

表 10-30 函数 FLASH_Half_Page_Write_Size_Config_RAM

函数名	FLASH_Half_Page_Write_Size_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Half_Page_Write_Size_Config_RAM (uint32_t WriteSize)
功能描述	配置 FLASH 半页编程模式下需要写入的字数。
输入参数 1	WriteSize: 需要写入的字数, 取值 6 位数值。
返回值	无
被调用函数 1	无

10.4.26 函数 FLASH_Program_Cmd_Config_RAM

表 10-31 函数 FLASH_Program_Cmd_Config_RAM

函数名	FLASH_Program_Cmd_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Program_Cmd_Config_RAM (uint32_t CmdSelect)
功能描述	配置 FLASH 命令选择。
输入参数 1	输入 CmdSelect: FLASH 命令选择, 取值为: FLASH_PROGRAM_CMD_WORD: 单字编程 FLASH_PROGRAM_CMD_ALL_CODE: 片擦 FLASH_PROGRAM_CMD_PAGE: 页擦 FLASH_PROGRAM_CMD_HALF_PAGE: 半页编程
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.27 函数 FLASH_Executive_Cmd_RAM

表 10-32 函数 FLASH_Executive_Cmd_RAM

函数名	FLASH_Executive_Cmd_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Executive_Cmd_RAM(void)
功能描述	执行当前 FLASH_ISPCMD 寄存器设定的命令, 即 FLASH_Program_Cmd_Config 函数配置的命令。
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.28 函数 FLASH_Executive_Cmd

表 10-33 函数 FLASH_Executive_Cmd

函数名	FLASH_Executive_Cmd
函数原型	void FLASH_Executive_Cmd(void)
功能描述	执行当前 FLASH_ISPCMD 寄存器设定的命令，即 FLASH_Program_Cmd_Config 函数配置的命令
输入参数 1	无
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.29 函数 FLASH_NonVolatile_Memory_ECC_Enable_RAM

表 10-34 函数 FLASH_NonVolatile_Memory_ECC_Enable_RAM

函数名	FLASH_NonVolatile_Memory_ECC_Enable_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_NonVolatile_Memory_ECC_Enable_RAM (FunctionalState NewState)
功能描述	配置 FLASH 非易失性存储器 ECC 使能。
输入参数 1	NewState: FLASH 非易失性存储器 ECC 使能状态，取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

10.4.30 函数 FLASH_Linear_Prefetch_Enable_RAM

表 10-35 函数 FLASH_Linear_Prefetch_Enable_RAM

函数名	FLASH_Linear_Prefetch_Enable_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Linear_Prefetch_Enable_RAM (FunctionalState NewState)
功能描述	配置 FLASH 线性预取使能。
输入参数 1	NewState: FLASH 线性预取使能状态，取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

10.4.31 函数 FLASH_Period_Number_Config_RAM

表 10-36 函数 FLASH_Period_Number_Config_RAM

函数名	FLASH_Period_Number_Config_RAM
-----	--------------------------------

函数原型	void __attribute__((section(".indata"))) FLASH_Period_Number_Config_RAM (uint32_t PeriodNum)
功能描述	配置 FLASH 访问周期。
输入参数 1	PeriodNum: FLASH 访问系统周期个数，取值为 1~16。
返回值	无
被调用函数 1	static inline uint32_t __attribute__((section(".indata"))) SFR_Config_RAM (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

10.4.32 函数 FLASH_Program_Addr_Config_RAM

表 10-37 函数 FLASH_Program_Addr_Config_RAM

函数名	FLASH_Program_Addr_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Program_Addr_Config_RAM (uint32_t ProgramAddr)
功能描述	配置 FLASH 编程和行擦期间的地址
输入参数 1	ProgramAddr: FLASH 编程地址，取值为 0x0~0xFFFFF，硬件忽略低 2 位。
返回值	无
被调用函数 1	无

10.4.33 函数 FLASH_Get_Program_Status_RAM

表 10-38 函数 FLASH_Get_Program_Status_RAM

函数名	FLASH_Get_Program_Status_RAM
函数原型	FlagStatus __attribute__((section(".indata"))) FLASH_Get_Program_Status_RAM (void)
功能描述	读 FLASH 编程状态。
输入参数 1	无
返回值	编程状态，1: FLASH 正处于编程状态，0: FLASH 不在编程状态。
被调用函数 1	无

10.4.34 函数 FLASH_Get_Program_Status

表 10-39 函数 FLASH_Get_Program_Status

函数名	FLASH_Get_Program_Status
函数原型	FlagStatus FLASH_Get_Program_Status (void)
功能描述	读 FLASH 编程状态。
输入参数 1	无
返回值	编程状态，1: FLASH 正处于编程状态，0: FLASH 不在编程状态。
被调用函数 1	无

10.4.35 函数 FLASH_Get_Wipe_Status_RAM

表 10-40 函数 FLASH_Get_Wipe_Status_RAM

函数名	FLASH_Get_Wipe_Status_RAM
函数原型	FlagStatus __attribute__((section(".indata"))) FLASH_Get_Wipe_Status_RAM (void)
功能描述	读 FLASH 擦写状态
输入参数 1	无
返回值	编程状态，1：ISP 正在执行擦除或者写命令，0：ISP 处于空闲状态。
被调用函数 1	无

10.4.36 函数 FLASH_Get_Wipe_Status

表 10-41 函数 FLASH_Get_Wipe_Status

函数名	FLASH_Get_Wipe_Status
函数原型	FlagStatus FLASH_Get_Wipe_Status (void)
功能描述	读 FLASH 擦写状态。
输入参数 1	无
返回值	编程状态，1：ISP 正在执行擦除或者写命令，0：ISP 处于空闲状态。
被调用函数 1	无

10.4.37 函数 FLASH_Get_Compute_Complete_Status_RAM

表 10-42 函数 FLASH_Get_Compute_Complete_Status_RAM

函数名	FLASH_Get_Compute_Complete_Status_RAM
函数原型	FlagStatus __attribute__((section(".indata"))) FLASH_Get_Compute_Complete_Status_RAM (void)
功能描述	读 FLASH 计算完成状态。
输入参数 1	无
返回值	编程状态，1：计算完成，0：未计算或者计算中。
被调用函数 1	无

10.4.38 函数 FLASH_Clear_Compute_Complete_Status_RAM

表 10-43 函数 FLASH_Clear_Compute_Complete_Status_RAM

函数名	FLASH_Clear_Compute_Complete_Status_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Clear_Compute_Complete_Status_RAM (void)
功能描述	清零 FLASH 计算完成状态。
输入参数 1	无

返回值	无
被调用函数 1	无

10.4.39 函数 FLASH_Get_CFG_Error_Flag_RAM

表 10-44 函数 FLASH_Get_CFG_Error_Flag_RAM

函数名	FLASH_Get_CFG_Error_Flag_RAM
函数原型	FlagStatus __attribute__((section(".indata"))) FLASH_Get_CFG_Error_Flag_RAM (void)
功能描述	读 FLASH CFG 编程错误标志
输入参数 1	无
返回值	编程错误标志，1：CFG 配置区编程错误，0：正常
被调用函数 1	无

10.4.40 函数 FLASH_Clear_CFG_Error_Flag_RAM

表 10-45 函数 FLASH_Clear_CFG_Error_Flag_RAM

函数名	FLASH_Clear_CFG_Error_Flag_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Clear_CFG_Error_Flag_RAM (void)
功能描述	清零 FLASH CFG 编程错误标志。
输入参数 1	无
返回值	无
被调用函数 1	无

10.4.41 函数 FLASH_CheckSum_Addr_Config_RAM

表 10-46 函数 FLASH_CheckSum_Addr_Config_RAM

函数名	FLASH_CheckSum_Addr_Config_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_CheckSum_Addr_Config_RAM(uint32_t StartAddr, uint32_t StopAddr)
功能描述	配置 FLASH CheckSum 首尾地址
输入参数 1	StartAddr: FLASH CheckSum 的首地址，地址区间为 0x0~0xFFFFF，硬件忽略低 4 位。
输入参数 2	StopAddr: FLASH CheckSum 的尾地址，地址区间为 0x0~0xFFFFF，硬件忽略低 4 位。
返回值	无
被调用函数 1	无

10.4.42 函数 FLASH_Start_SIG_Compute_Enable_RAM

表 10-47 函数 FLASH_Start_SIG_Compute_Enable_RAM

函数名	FLASH_Start_SIG_Compute_Enable_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Start_SIG_Compute_Enable_RAM (FunctionalState NewState)
功能描述	配置 FLASH 启动 SIG 计算使能。
输入参数 1	NewState: FLASH 启动 SIG 计算使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

10.4.43 函数 FLASH_Get_CheckSum_Result_RAM

表 10-48 函数 FLASH_Get_CheckSum_Result_RAM

函数名	FLASH_Get_CheckSum_Result_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Get_CheckSum_Result_RAM (FLASH_CheckSumResult* CheckSumStruct)
功能描述	获取 FLASH CheckSum 的结果。
输入参数 1	CheckSumStruct: FLASH CheckSum 信息结构体指针, 指针必须指向可写的地址空间。
返回值	无
被调用函数 1	无

10.4.44 函数 FLASH_Wipe_Configuration_RAM

表 10-49 函数 FLASH_Wipe_Configuration_RAM

函数名	FLASH_Wipe_Configuration_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Wipe_Configuration_RAM (uint32_t WipeMode, uint32_t WipeAddr)
功能描述	FLASH 擦除操作配置, 用户需要确认 FLASH 擦写空闲状态。
输入参数 1	WipeMode: FLASH 擦除模式, 取值为: FLASH_WIPE_CODE_PAGE: 程序区页擦 FLASH_WIPE_CFG_PAGE: 用户配置区页擦 FLASH_WIPE_CODE_ALL: 程序区片擦
输入参数 2	WipeAddr: 擦除地址, 取值为 0x0~0xFFFFF, 硬件忽略低 10 位。
返回值	无
被调用函数 1	FLASH_Unlock_ISP_RAM
被调用函数 2	FLASH_Unlock_Code_RAM
被调用函数 3	FLASH_Unlock_User_Config_RAM
被调用函数 4	SFR_Config_RAM

被调用函数 5	FLASH_Executive_Cmd_RAM
被调用函数 6	FLASH_Get_Wipe_Status_RAM
被调用函数 7	FLASH_Zone_Config_RAM
被调用函数 8	FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM
被调用函数 9	FLASH_Clear_Flash_Unlock_Status_RAM
被调用函数 10	FLASH_Clear_Config_Unlock_Status_RAM

10.4.45 函数 FLASH_Wipe_Configuration

表 10-50 函数 FLASH_Wipe_Configuration

函数名	FLASH_Wipe_Configuration
函数原型	void FLASH_Wipe_Configuration (uint32_t WipeMode,uint32_t WipeAddr)
功能描述	FLASH 擦除操作配置，用户需要确认 FLASH 擦写空闲状态。
输入参数 1	WipeMode: FLASH 擦除模式，取值为： FLASH_WIPE_CODE_PAGE: 程序区页擦 FLASH_WIPE_CFG_PAGE: 用户配置区页擦 FLASH_WIPE_CODE_ALL: 程序区片擦
输入参数 2	WipeAddr: 擦除地址，取值为 0x0~0xFFFFF，硬件忽略低 10 位。
返回值	无
被调用函数 1	FLASH_Unlock_ISP
被调用函数 2	FLASH_Unlock_Code
被调用函数 3	FLASH_Unlock_User_Config
被调用函数 4	SFR_Config
被调用函数 5	FLASH_Executive_Cmd
被调用函数 6	FLASH_Get_Wipe_Status
被调用函数 7	FLASH_Zone_Config_RAM
被调用函数 8	FLASH_Clear_NonVolatile_Memory_Unlock_Status
被调用函数 9	FLASH_Clear_Flash_Unlock_Status
被调用函数 10	FLASH_Clear_Config_Unlock_Status

10.4.46 函数 FLASH_Program_Configuration_RAM

表 10-51 函数 FLASH_Program_Configuration_RAM

函数名	FLASH_Program_Configuration_RAM
函数原型	void __attribute__((section(".indata"))) FLASH_Program_Configuration_RAM (FLASH_ProgramTypeDef * flashProgramStruct)
功能描述	FLASH 编程操作配置，用户需要确认 FLASH 擦写空闲状态
输入参数 1	flashProgramStruct: FLASH 编程信息结构体指针。
返回值	无
被调用函数 1	FLASH_Unlock_ISP_RAM

被调用函数 2	FLASH_Unlock_User_Config_RAM
被调用函数 3	FLASH_Unlock_Code_RAM
被调用函数 4	SFR_Config_RAM
被调用函数 5	FLASH_Executive_Cmd_RAM
被调用函数 6	FLASH_Get_Program_Status_RAM
被调用函数 7	FLASH_Get_Wipe_Status_RAM
被调用函数 8	FLASH_Zone_Config_RAM
被调用函数 9	FLASH_Data_Write_Enable_RAM
被调用函数 10	FLASH_Clear_NonVolatile_Memory_Unlock_Status_RAM
被调用函数 11	FLASH_Clear_Flash_Unlock_Status_RAM
被调用函数 12	FLASH_Clear_Config_Unlock_Status_RAM

10.4.47 函数 FLASH_Program_Configuration

表 10-52 函数 FLASH_Program_Configuration

函数名	FLASH_Program_Configuration
函数原型	void FLASH_Program_Configuration (FLASH_ProgramTypeDef * flashProgramStruct)
功能描述	FLASH 编程操作配置，用户需要确认 FLASH 擦写空闲状态。
输入参数 1	flashProgramStruct: FLASH 编程信息结构体指针。
返回值	无
被调用函数 1	FLASH_Unlock_ISP
被调用函数 2	FLASH_Unlock_User_Config
被调用函数 3	FLASH_Unlock_Code
被调用函数 4	SFR_Config
被调用函数 5	FLASH_Executive_Cmd
被调用函数 6	FLASH_Get_Program_Status
被调用函数 7	FLASH_Get_Wipe_Status
被调用函数 8	FLASH_Zone_Config
被调用函数 9	FLASH_Data_Write_Enable_RAM
被调用函数 10	FLASH_Clear_NonVolatile_Memory_Unlock_Status
被调用函数 11	FLASH_Clear_Flash_Unlock_Status
被调用函数 12	FLASH_Clear_Config_Unlock_Status

10.4.48 函数 Read_Flash_or_CFR_RAM

表 10-53 函数 Read_Flash_or_CFR_RAM

函数名	Read_Flash_or_CFR_RAM
函数原型	uint32_t __attribute__((section(".indata"))) Read_Flash_or_CFR_RAM (uint32_t address,uint32_t ZoneSelect)

功能描述	读取 flash 的程序区或者信息区(RAM)
输入参数 1	address(读取地址)
输入参数 2	ZoneSelect (读取区域选择)
返回值	输入指向地址的 32 位数据
被调用函数 1	FLASH_Unlock_ISP_RAM
被调用函数 2	FLASH_Unlock_User_Config_RAM
被调用函数 3	FLASH_Unlock_Code_RAM
被调用函数 4	FLASH_Zone_Config_RAM
被调用函数 5	FLASH_Clear_NonVolatile_Memory_Unlock_Status
被调用函数 6	FLASH_Clear_Flash_Unlock_Status
被调用函数 7	FLASH_Clear_Config_Unlock_Status

10.4.49 函数 Read_Flash_or_CFR

表 10-54 函数 Read_Flash_or_CFR

函数名	Read_Flash_or_CFR
函数原型	uint32_t Read_Flash_or_CFR (uint32_t address,uint32_t ZoneSelect)
功能描述	读取 flash 的程序区或者信息区
输入参数 1	address(读取地址)
输入参数 2	ZoneSelect (读取区域选择)
返回值	输入指向地址的 32 位数据
被调用函数 1	FLASH_Unlock_ISP_RAM
被调用函数 2	FLASH_Unlock_User_Config_RAM
被调用函数 3	FLASH_Unlock_Code_RAM
被调用函数 4	FLASH_Zone_Config_RAM
被调用函数 5	FLASH_Clear_NonVolatile_Memory_Unlock_Status
被调用函数 6	FLASH_Clear_Flash_Unlock_Status
被调用函数 7	FLASH_Clear_Config_Unlock_Status

10.4.50 函数 Read_Soft_Device_ID1

表 10-55 函数 Read_Soft_Device_ID1

函数名	Read_Soft_Device_ID1
函数原型	uint32_t Read_Soft_Device_ID1(void)
功能描述	读取芯片唯一 ID
输入参数 1	无
返回值	对应的唯一 ID
被调用函数 1	无

10.4.51 函数 Read_Soft_Device_ID2

表 10-56 函数 Read_Soft_Device_ID2

函数名	Read_Soft_Device_ID2
函数原型	uint32_t Read_Soft_Device_ID2(void)
功能描述	读取芯片唯一 ID
输入参数 1	无
返回值	对应的唯一 ID
被调用函数 1	无

10.4.52 函数 Read_Soft_Device_ID3

表 10-57 函数 Read_Soft_Device_ID3

函数名	Read_Soft_Device_ID3
函数原型	uint32_t Read_Soft_Device_ID3(void)
功能描述	读取芯片唯一 ID
输入参数 1	无
返回值	对应的唯一 ID
被调用函数 1	无

10.4.53 函数 Read_Soft_Device_ID4

表 10-58 函数 Read_Soft_Device_ID4

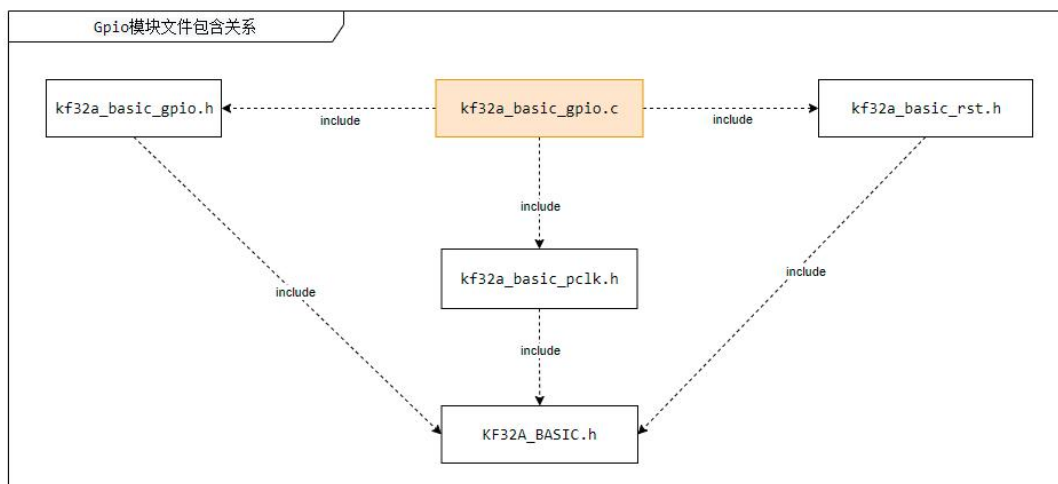
函数名	Read_Soft_Device_ID4
函数原型	uint32_t Read_Soft_Device_ID4(void)
功能描述	读取芯片唯一 ID
输入参数 1	无
返回值	对应的唯一 ID
被调用函数 1	无

11 通用输入/输出引脚（GPIO）

KungFu32 内核提供了一个 24 位的系统节拍定时器（System Tick Timer）。系统节拍定时器可为系统提供可编程时长的周期性中断，能工作在休眠模式下。

11.1 文件引用关系

图 11-1 GPIO 模块文件包含关系



11.2 GPIO 寄存器结构

GPIO 寄存器结构，GPIO_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct GPIO_MemMap
{
    volatile const uint32_t    PIR;
    volatile uint32_t          POR;
    volatile uint32_t          PUR;
    volatile uint32_t          PDR;
    volatile uint32_t          PODR;
    volatile uint32_t          PMOD;
    volatile uint32_t          OMOD;
    volatile uint32_t          LOCK;
    volatile uint32_t          RMP[2];
    volatile uint32_t          RESERVED[3];
    volatile uint32_t          RMP_MSB;
}GPIO_SFRmap;
    
```

表 11-1 GPIO 寄存器结构说明

寄存器	描述
PIR	GPIO 输入状态寄存器, 偏移:0x0
POR	GPIO 输出状态寄存器, 偏移:0x4
PUR	GPIO 上拉使能寄存器, 偏移:0x8
PDR	GPIO 下拉使能寄存器, 偏移:0xC
PODR	GPIO 开漏输出控制寄存器, 偏移:0x10
PMOD	GPIO 端口方向控制寄存器, 偏移:0x14
OMOD	GPIO 端口速度控制寄存器, 偏移:0x18
LOCK	GPIO 端口锁定寄存器, 偏移:0x1C
RMP[2]	GPIO 重映射控制寄存器, 偏移:0x20
RESERVED[3]	偏移:0x2C
RMP_MSB	GPIO 触摸端口配置寄存器, 偏移 0x30

GPIO 配置信息结构体, GPIO_MemMap, 定义于文件 kf32a_basic_systick.h 中, 如下:

```
typedef struct
{
    uint32_t    m_Pin;
    GPIOMode_TypeDef    m_Mode;
    GPIOSpeed_TypeDef    m_Speed;
    GPIOPOD_TypeDef    m_OpenDrain;
    GPIOPU_TypeDef    m_PullUp;
    GPIOPD_TypeDef    m_PullDown;
} GPIO_InitTypeDef;
```

表 11-2 GPIO 配置信息结构体说明

寄存器	描述
m_Pin	GPIO 端口掩码, 取值为宏“GPIO 端口掩码”中的一个或多个组合
m_Mode	GPIO 输出模式, 取值为枚举类型 GPIOMode_TypeDef 中的一个
m_Speed	GPIO 输出速度, 取值为枚举类型 GPIOSpeed_TypeDef 中的一个
m_OpenDrain	GPIO 开漏控制, 取值为枚举类型 GPIOPOD_TypeDef 中的一个
m_PullUp	GPIO 上拉配置, 取值为枚举类型 GPIOPU_TypeDef 中的一个
m_PullDown	GPIO 下拉配置, 取值为枚举类型 GPIOPD_TypeDef 中的一个

11.3 GPIO 宏定义

GPIO 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, GPIO 其他相关宏定义详见文件 kf32a_basic_gpio.h 及函数参数描述。

11.4 GPIO 库函数

表 11-3 GPIO 固件库函数列表

函数名	描述
GPIO_Reset	复位 GPIO 外设，使能外设时钟
GPIO_Configuration	通用输入输出端口(GPIO)初始化配置
GPIO_Struct_Init	初始化 GPIO 配置信息结构体
GPIO_Pin_Lock_Config	配置 GPIO 端口锁定
GPIO_Pull_Up_Enable	配置 GPIO 端口上拉使能
GPIO_Toggle_Pull_Up_Config	配置 GPIO 端口上拉使能取反
GPIO_Pull_Down_Enable	配置 GPIO 端口下拉使能
GPIO_Toggle_Pull_Down_Config	配置 GPIO 端口下拉使能取反
GPIO_Open_Drain_Enable	配置 GPIO 端口开漏输出控制
GPIO_Toggle_Open_Drain_Config	配置 GPIO 端口开漏输出控制取反
GPIO_Write_Mode_Bits	配置 GPIO 输出模式控制
GPIO_Speed_Config	配置 GPIO 端口输出速度控制
GPIO_Toggle_Speed_Config	配置 GPIO 端口输出速度控制取反
GPIO_Read_Input_Data_Bit	获取 GPIO 端口指定引脚的输入数据
GPIO_Read_Input_Data	获取 GPIO 端口输入数据
GPIO_Read_Output_Data_Bit	获取 GPIO 端口指定引脚的输出数据
GPIO_Read_Output_Data	获取 GPIO 端口输出数据
GPIO_Set_Output_Data_Bits	GPIO_Set_Output_Data_Bits
GPIO_Toggle_Output_Data_Config	配置 GPIO 端口指定引脚的输出数据取反
GPIO_Pin_RMP_Config	配置 GPIO 端口引脚重映射

11.4.1 函数 GPIO_Reset

表 11-4 函数 GPIO_Reset

函数名	GPIO_Reset
函数原型	void GPIO_Reset (GPIO_SFRmap* GPIOx)
功能描述	复位 GPIO 外设，使能外设时钟
输入参数 1	输入 GPIOx: 指向 GPIO 内存结构的指针，取值为 GPIOA_SFR~GPIOH_SFR
返回值	无
被调用函数	无

11.4.2 函数 GPIO_Configuration

表 11-5 函数 GPIO_Configuration

函数名	GPIO_Configuration
-----	--------------------

函数原型	void GPIO_Configuration (GPIO_SFRmap* GPIOx, GPIO_InitTypeDef* gpioInitStruct)
功能描述	通用输入输出端口(GPIO)初始化配置
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR。
输入参数 2	gpioInitStruct: GPIO 配置信息结构体指针
返回值	无
被调用函数	无

11.4.3 函数 GPIO_Struct_Init

表 11-6 函数 GPIO_Struct_Init

函数名	GPIO_Struct_Init
函数原型	void GPIO_Struct_Init (GPIO_InitTypeDef* gpioInitStruct)
功能描述	初始化 GPIO 配置信息结构体
输入参数 1	gpioInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

11.4.4 函数 GPIO_Pin_Lock_Config

表 11-7 函数 GPIO_Pin_Lock_Config

函数名	GPIO_Pin_Lock_Config
函数原型	void GPIO_Pin_Lock_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin, FunctionalState NewState)
功能描述	配置 GPIO 端口锁定
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合。
输入参数 3	NewState: GPIO 端口锁定状态, 取值范围为 TRUE 或 FALSE
返回值	无
被调用函数	无

11.4.5 函数 GPIO_Pull_Up_Enable

表 11-8 函数 GPIO_Pull_Up_Enable

函数名	GPIO_Pull_Up_Enable
函数原型	Void GPIO_Pull_Up_Enable (GPIO_SFRmap* GPIOx,uint16_t GpioPin, FunctionalState NewState)
功能描述	配置 GPIO 端口上拉使能
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR

输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的 一个或多个组合。
输入参数 3	NewState: GPIO 端口上拉使能状态, 取值范围为 TRUE 或 FALSE
返回值	无
被调用函数	无

11.4.6 函数 GPIO_Toggle_Pull_Up_Config

表 11-9 函数 GPIO_Toggle_Pull_Up_Config

函数名	GPIO_Toggle_Pull_Up_Config
函数原型	void GPIO_Toggle_Pull_Up_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	配置 GPIO 端口上拉使能取反
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
返回值	无
被调用函数	无

11.4.7 函数 GPIO_Pull_Down_Enable

表 11-10 函数 GPIO_Pull_Down_Enable

函数名	GPIO_Pull_Down_Enable
函数原型	void GPIO_Pull_Down_Enable (GPIO_SFRmap* GPIOx,uint16_t GpioPin, FunctionalState NewState)
功能描述	配置 GPIO 端口下拉使能
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
输入参数 3	NewState: GPIO 端口下拉使能状态, 取值范围为 TRUE 或 FALSE
返回值	无
被调用函数	无

11.4.8 函数 GPIO_Toggle_Pull_Down_Config

表 11-11 函数 GPIO_Toggle_Pull_Down_Config

函数名	GPIO_Toggle_Pull_Down_Config
函数原型	void GPIO_Toggle_Pull_Down_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	配置 GPIO 端口下拉使能取反

输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
返回值	无
被调用函数	无

11.4.9 函数 GPIO_Open_Drain_Enable

表 11-12 函数 GPIO_Open_Drain_Enable

函数名	GPIO_Open_Drain_Enable
函数原型	Void GPIO_Open_Drain_Enable (GPIO_SFRmap* GPIOx, uint16_t GpioPin, GPIOPOD_TypeDef NewState)
功能描述	配置 GPIO 端口开漏输出控制
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
输入参数 3	NewState: GPIO 开漏输出控制状态, 取值为: GPIO_POD_PP: 推挽输出, GPIO_POD_OD: 开漏输出
返回值	无
被调用函数	无

11.4.10 函数 GPIO_Toggle_Open_Drain_Config

表 11-13 函数 GPIO_Toggle_Open_Drain_Config

函数名	GPIO_Toggle_Open_Drain_Config
函数原型	void GPIO_Toggle_Open_Drain_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	配置 GPIO 端口开漏输出控制取反
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合。
返回值	无
被调用函数	无

11.4.11 函数 GPIO_Write_Mode_Bits

表 11-14 函数 GPIO_Write_Mode_Bits

函数名	GPIO_Write_Mode_Bits
函数原型	void GPIO_Write_Mode_Bits (GPIO_SFRmap* GPIOx, uint16_t GpioPin, GPIOMode_TypeDef NewState)

功能描述	配置 GPIO 输出模式控制
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
输入参数 3	NewState: GPIO 输出模式控制状态, 取值为: GPIO_MODE_IN: 通用 IO 口输入模式 GPIO_MODE_OUT: 通用 IO 口输出模式 GPIO_MODE_RMP: 重映射 IO 口功能模式 GPIO_MODE_AN: 模拟模式
返回值	无
被调用函数	无

11.4.12 函数 GPIO_Speed_Config

表 11-15 函数 GPIO_Speed_Config

函数名	GPIO_Speed_Config
函数原型	Void GPIO_Speed_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin, GPIO_Speed_TypeDef NewState)
功能描述	配置 GPIO 端口输出速度控制
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
输入参数 3	NewState: GPIO 端口输出速度控制状态, 取值为: GPIO_LOW_SPEED: 10MHZ GPIO_HIGH_SPEED: 50MHZ
返回值	无
被调用函数	无

11.4.13 函数 GPIO_Toggle_Speed_Config

表 11-16 函数 GPIO_Toggle_Speed_Config

函数名	GPIO_Toggle_Speed_Config
函数原型	void GPIO_Toggle_Speed_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	配置 GPIO 端口输出速度控制取反
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
返回值	无
被调用函数	无

11.4.14 函数 GPIO_Read_Input_Data_Bit

表 11-17 函数 GPIO_Read_Input_Data_Bit

函数名	GPIO_Read_Input_Data_Bit
函数原型	void GPIO_Read_Input_Data_Bit (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	通用输入输出端口(GPIO)初始化配置
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 指定端口引脚, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个
返回值	指定引脚的信息, 1: 逻辑高电平, 0: 逻辑低电平
被调用函数	无

11.4.15 函数 GPIO_Read_Input_Data

表 11-18 函数 GPIO_Read_Input_Data

函数名	GPIO_Read_Input_Data
函数原型	void GPIO_Read_Input_Data (GPIO_SFRmap* GPIOx)
功能描述	获取 GPIO 端口输入数据
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
返回值	输入数据, 16 位有效数据
被调用函数	无

11.4.16 函数 GPIO_Read_Output_Data_Bit

表 11-19 函数 GPIO_Read_Output_Data_Bit

函数名	GPIO_Read_Output_Data_Bit
函数原型	void GPIO_Read_Output_Data_Bit (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	获取 GPIO 端口指定引脚的输出数据
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR
输入参数 2	GpioPin: 指定端口引脚, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个
返回值	指定引脚的信息, 1: 逻辑高电平, 0: 逻辑低电平
被调用函数	无

11.4.17 函数 GPIO_Read_Output_Data

表 11-20 函数 GPIO_Read_Output_Data

函数名	GPIO_Read_Output_Data
-----	-----------------------

函数原型	void GPIO_Read_Output_Data (GPIO_SFRmap* GPIOx)
功能描述	获取 GPIO 端口输出数据
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR。
返回值	输出数据, 16 位有效数据
被调用函数	无

11.4.18 函数 GPIO_Set_Output_Data_Bits

表 11-21 函数 GPIO_Set_Output_Data_Bits

函数名	GPIO_Set_Output_Data_Bits
函数原型	void GPIO_Set_Output_Data_Bits (GPIO_SFRmap* GPIOx, uint16_t GpioPin, BitAction BitsValue)
功能描述	配置 GPIO 端口指定引脚的输出数据
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR。
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
输入参数 3	BitsValue: 引脚的配置值, 取值为: Bit_RESET: 逻辑低电平 Bit_SET: 逻辑高电平
返回值	无
被调用函数	无

11.4.19 函数 GPIO_Toggle_Output_Data_Config

表 11-22 函数 GPIO_Toggle_Output_Data_Config

函数名	GPIO_Toggle_Output_Data_Config
函数原型	void GPIO_Toggle_Output_Data_Config (GPIO_SFRmap* GPIOx, uint16_t GpioPin)
功能描述	配置 GPIO 端口指定引脚的输出数据取反
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR。
输入参数 2	GpioPin: 端口引脚掩码, 取值为 GPIO_PIN_MASK_0~GPIO_PIN_MASK_15 中的一个或多个组合
返回值	无
被调用函数	无

11.4.20 函数 GPIO_Pin_RMP_Config

表 11-23 函数 GPIO_Pin_RMP_Config

函数名	GPIO_Pin_RMP_Config
函数原型	void GPIO_Pin_RMP_Config (GPIO_SFRmap* GPIOx,

	uint16_t GpioPinNum, uint8_t PinRemap)
功能描述	配置 GPIO 端口引脚重映射
输入参数 1	GPIOx: 指向 GPIO 内存结构的指针, 取值为 GPIOA_SFR~GPIOH_SFR。
输入参数 2	GpioPinNum: 指定端口引脚, 取值为 GPIO_Pin_Num_0~GPIO_Pin_Num_15 中的一个
输入参数 3	<p>PinRemap: 引脚重映射选择, 取值为:</p> <p>GPIO_RMP_AF0_SYSTEM: SYSTEM</p> <p>GPIO_RMP_AF1_T0: T0</p> <p>GPIO_RMP_AF1_T1: T1</p> <p>GPIO_RMP_AF1_T2: T2</p> <p>GPIO_RMP_AF1_T3: T3</p> <p>GPIO_RMP_AF1_T4: T4</p> <p>GPIO_RMP_AF2_T5: T5</p> <p>GPIO_RMP_AF2_T6: T6</p> <p>GPIO_RMP_AF2_T9: T9</p> <p>GPIO_RMP_AF2_T10: T10</p> <p>GPIO_RMP_AF3_T9: T9</p> <p>GPIO_RMP_AF3_T20: T20</p> <p>GPIO_RMP_AF3_T21: T21</p> <p>GPIO_RMP_AF3_T23: T23</p> <p>GPIO_RMP_AF3_QEI1: QEI1</p> <p>GPIO_RMP_AF4_T: T9</p> <p>GPIO_RMP_AF4_T14: T14</p> <p>GPIO_RMP_AF4_T15: T15</p> <p>GPIO_RMP_AF4_T18: T18</p> <p>GPIO_RMP_AF4_T19: T19</p> <p>GPIO_RMP_AF4_T22: T22</p> <p>GPIO_RMP_AF4_QEI0: QEI0</p> <p>GPIO_RMP_AF5_USART0: USART0</p> <p>GPIO_RMP_AF5_USART1: USART1</p> <p>GPIO_RMP_AF5_USART2: USART2</p> <p>GPIO_RMP_AF6_USART3: USART3</p> <p>GPIO_RMP_AF6_USART4: USART4</p> <p>GPIO_RMP_AF6_USART5: USART5</p> <p>GPIO_RMP_AF6_USART6: USART6</p> <p>GPIO_RMP_AF6_USART7: USART7</p> <p>GPIO_RMP_AF7_SPI0: SPI0</p> <p>GPIO_RMP_AF7_SPI1: SPI1</p> <p>GPIO_RMP_AF7_SPI2: SPI2</p> <p>GPIO_RMP_AF7_SPI3: SPI3</p> <p>GPIO_RMP_AF8_I2C0: I2C0</p> <p>GPIO_RMP_AF8_I2C1: I2C1</p>

	GPIO_RMP_AF8_I2C2: I2C2 GPIO_RMP_AF8_I2C3: I2C3 GPIO_RMP_AF9_CAN0: CAN0 GPIO_RMP_AF9_CAN1: CAN1 GPIO_RMP_AF9_CAN2: CAN2 GPIO_RMP_AF9_CAN3: CAN3 GPIO_RMP_AF9_CAN4: CAN4 GPIO_RMP_AF9_CAN5: CAN5 GPIO_RMP_AF9_FLT: FLT GPIO_RMP_AF10_CCP: CCP GPIO_RMP_AF10_SPI: SPI GPIO_RMP_AF10_I2C: I2C GPIO_RMP_AF11_USART: USART GPIO_RMP_AF11_CCP: CCP GPIO_RMP_AF11_SPI: SPI GPIO_RMP_AF11_I2C: I2C GPIO_RMP_AF12_LED: LED GPIO_RMP_AF12_CFGL: CFGL GPIO_RMP_AF13_EXIC: EXIC GPIO_RMP_AF14_LED: LED GPIO_RMP_AF15_TESTPAD: TESTPAD
返回值	无
被调用函数	无

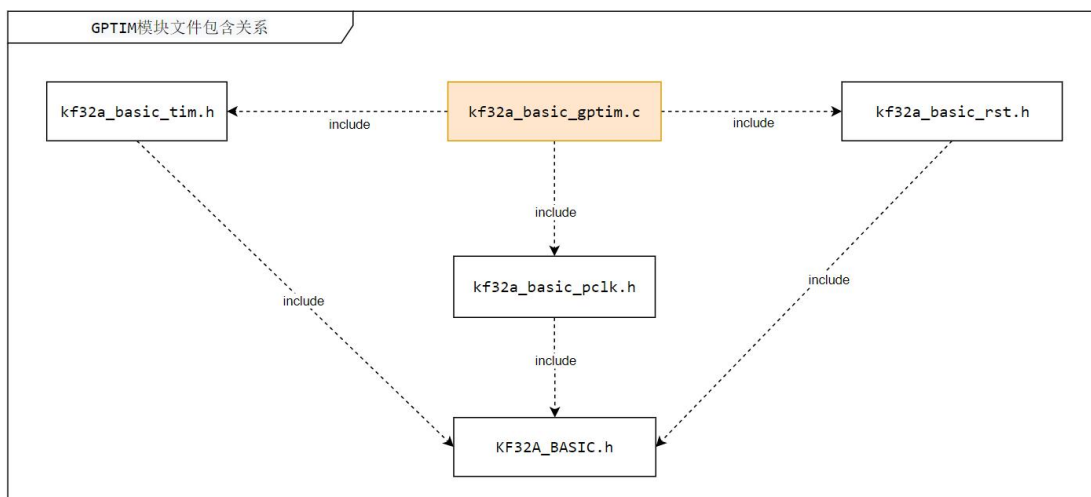
12 通用定时/计数器（GPTIM）

Tx(x=0,1,2,3,4,18,19,22,23)是 16 位的定时/计数器，Tx(x=20,21)是 32 位的定时/计数器。它们除位宽不一样外，其他功能以及实现方式都是一样的。其中 T0 可作为低功耗定时器使用。

通用定时/计数器有定时和计数 2 种工作模式，支持 3 种计数方式：向上计数、向下计数和向上向下计数方式。根据不同的模式，计数会产生溢出，将 Tx 中断标志位 TXIF 置 1。Tx 属于外部单元，因此在使用 Tx 中断时，需使能对应的外设中断。

12.1 文件引用关系

图 12-1 GPTIM 模块文件包含关系



12.2 GPTIM 寄存器结构

GPTIM 寄存器结构，GPTIM_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct GPTIM_MemMap
{
    volatile uint32_t    CNT;
    volatile uint32_t    CTL1;
    volatile uint32_t    CTL2;
    volatile uint32_t    PRSC;
    volatile uint32_t    PPX;
    volatile uint32_t    UDTIM;
    uint32_t    RESERVED1[2];
    volatile const uint32_t    CCPXC1;
    volatile const uint32_t    CCPXC2;
}
    
```

```
volatile const uint32_t    CCPXC3;
volatile const uint32_t    CCPXC4;
volatile uint32_t    CCPXSRIC;
volatile const uint32_t    CCPXDF;
uint32_t    RESERVED2[2];
volatile uint32_t    CCPXCTL1;
volatile uint32_t    CCPXR1;
volatile uint32_t    CCPXR2;
volatile uint32_t    CCPXR3;
volatile uint32_t    CCPXR4;
volatile uint32_t    CCPXCTL2;
volatile uint32_t    CCPXCTL3;
volatile uint32_t    CCPXEGIF;
} GPTIM_SFRmap;
```

表 12-1 GPTIM 寄存器结构说明

寄存器	描述
CNT	TxCNT 寄存器, 偏移:0x00
CTL1	控制寄存器 1, 偏移:0x04
CTL2	控制寄存器 2, 偏移:0x08
PRSC	Tx 预分频寄存器, 偏移:0x0C
PPX	Tx_PPx 周期寄存器, 偏移:0x10
UDTIM	更新计数器, 偏移:0x14
RESERVED1[2]	保留地址, 偏移:0x18
CCPXC1	CCP 捕捉寄存器 1, 偏移:0x20
CCPXC2	CCP 捕捉寄存器 2, 偏移:0x24
CCPXC3	CCP 捕捉寄存器 3, 偏移:0x28
CCPXC4	CCP 捕捉寄存器 4, 偏移:0x2C
CCPXSRI	CCP 中断标志清除寄存器, 偏移:0x30
CCPXDF	CCP 触发 DMA 中断标志寄存器, 偏移:0x34
RESERVED2[2]	保留地址, 偏移:0x3C
CCPXCTL1	CCP 控制寄存器 1, 偏移:0x40
CCPXR1	CCP 比较/PWM 占空比寄存器 1, 偏移:0x44
CCPXR2	CCP 比较/PWM 占空比寄存器 2, 偏移:0x48
CCPXR3	CCP 比较/PWM 占空比寄存器 3, 偏移:0x4C
CCPXR4	CCP 比较/PWM 占空比寄存器 4, 偏移:0x50
CCPXCTL2	CCP 控制寄存器 2, 偏移:0x54
CCPXCTL3	CCP 控制寄存器 3, 偏移:0x58
CCPXEGIF	CCPx 中断状态/事件产生寄存器, 偏移:0x5C

GPTIM 配置信息结构体, 定义于文件 kf32a_basic_tim.h 中, 如下:

```
typedef struct
```

```
{
    uint32_t    m_Counter;
    uint32_t    m_Period;
    uint32_t    m_Prescaler;
    uint16_t    m_CounterMode;
    uint16_t    m_Clock;
    uint16_t    m_WorkMode;
    uint16_t    m_MasterMode;
    uint16_t    m_SlaveMode;
    uint16_t    m_EXPulseSync;
    uint16_t    m_MasterSlaveSync;
} GPTIM_InitTypeDef;
```

表 12-2 GPTIM 配置信息结构体说明

配置信息	描述
m_Counter	定时器计数值，取值 32 位数据。
m_Period	定时器周期值，取值 32 位数据。
m_Prescaler	定时器预分频值，取值 32 位数据。
m_CounterMode	定时器计数模式，取值为宏“GPTIM 定时器计数模式”中的一个
m_Clock	定时器工作时钟，取值为宏“GPTIM 定时器工作时钟”中的一个
m_WorkMode	定时/计数模式选择，取值为宏“GPTIM 定时/计数模式选择”中的一个
m_MasterMode	主模式选择，取值为宏“GPTIM 主模式选择”中的一个
m_SlaveMode	从模式选择，取值为宏“GPTIM 从模式选择”中的一个
m_EXPulseSync	Tx 计数模式外部触发脉冲输入同步控制，取值为宏“GPTIM 计数模式外部触发脉冲输入同步控制”中的一个
m_MasterSlaveSync	主从模式同步控制，取值为 TRUE 或 FALSE

12.3 GPTIM 宏定义

GPTIM 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, GPTIM 其他相关宏定义详见文件 kf32a_basic_tim.h 及函数参数描述。

12.4 GPTIM 库函数

表 12-3 GPTIM 固件库函数列表

序号	函数名	描述
1	GPTIM_Configuration	通用定时器(GPTIM)配置。
2	GPTIM_Struct_Init	初始化通用定时器配置信息结构体。
3	GPTIM_Cmd	定时器启动控制使能。

4	GPTIM_Set_Counter	更新定时器计数值。
5	GPTIM_Set_Period	更新定时器周期值。
6	GPTIM_Set_Prescaler	更新定时器预分频值。
7	GPTIM_Counter_Mode_Config	更新定时器计数模式。
8	GPTIM_Clock_Config	更新定时器工作时钟。
9	GPTIM_External_Pulse_Sync_Config	更新定时器计数模式外部触发脉冲输入同步控制位。
10	GPTIM_Work_Mode_Config	更新定时/计数模式选择。
11	GPTIM_Update_Immediately_Config	配置立即更新控制位。
12	GPTIM_Master_Slave_Sync_Config	配置主从模式同步位。
13	GPTIM_Trigger_Select_Config	配置触发选择位。
14	GPTIM_Slave_Mode_Config	配置从模式选择位。
15	GPTIM_Master_Mode_Config	配置主模式选择位。
16	GPTIM_Update_Rising_Edge_Config	配置上升沿更新事件控制位。
17	GPTIM_Update_Enable	配置更新使能。
18	GPTIM_Trigger_DMA_Enable	允许触发事件的 DMA 请求配置
19	GPTIM_Update_DMA_Enable	配置更新事件的 DMA 请求使能。
20	GPTIM_Update_INT_Enable	配置 Tx 更新事件中断使能。
21	GPTIM_Trigger_INT_Enable	配置 Tx 触发事件中断使能。
22	GPTIM_Generate_Trigger_Config	产生触发事件配置位。
23	GPTIM_Get_Direction	读 TX 计数方向。
24	GPTIM_Get_Counter	读定时器计数值。
25	GPTIM_Get_Period	读定时器周期值。
26	GPTIM_Get_Prescaler	读定时器预分频值。
27	GPTIM_Overflow_INT_Enable	读取 Tx 计数溢出中断使能。
28	GPTIM_Clear_Overflow_INT_Flag	清除 Tx 溢出中断标志。
29	GPTIM_Clear_Update_INT_Flag	清除 Tx 更新事件中断标志。
30	GPTIM_Clear_Trigger_INT_Flag	清除 Tx 触发事件中断标志。
31	GPTIM_Get_Overflow_INT_Flag	读取 Tx 计数溢出中断标志。
32	GPTIM_Get_Update_INT_Flag	读取 Tx 更新事件中断标志。
33	GPTIM_Get_Trigger_INT_Flag	读取 Tx 触发事件中断标志。
34	GPTIM_Get_Update_DMA_INT_Flag	读取 Tx 更新事件触发 DMA 中断标志。
35	GPTIM_Get_Trigger_DMA_INT_Flag	配置触发事件触发 DMA 中断标志。

12.4.1 函数 GPTIM_Configuration

表 12-4 函数 GPTIM_Configuration

函数名	GPTIM_Configuration
函数原型	void GPTIM_Configuration(GPTIM_SFRmap*GPTIMx, GPTIM_InitTypeDef* gptimInitStruct)

功能描述	通用定时器(GPTIM)配置。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	gptimInitStruct: 通用定时器配置信息结构体指针。
返回值	无
被调用函数	无

12.4.2 函数 GPTIM_Struct_Init

表 12-5 函数 GPTIM_Struct_Init

函数名	GPTIM_Struct_Init
函数原型	void GPTIM_Struct_Init (GPTIM_InitTypeDef* gptimInitStruct)
功能描述	初始化通用定时器配置信息结构体。
输入参数 1	gptimInitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

12.4.3 函数 GPTIM_Cmd

表 12-6 函数 GPTIM_Cmd

函数名	GPTIM_Cmd
函数原型	void GPTIM_Cmd (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	定时器启动控制使能。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewState: 定时器使能控制, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

12.4.4 函数 GPTIM_Set_Counter

表 12-7 函数 GPTIM_Set_Counter

函数名	GPTIM_Set_Counter
函数原型	void GPTIM_Set_Counter (GPTIM_SFRmap* GPTIMx, uint32_t Counter)
功能描述	更新定时器计数值。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。

输入参数 2	Counter: 新的计数值, 取值 16 位数据。
返回值	无
被调用函数	无

12.4.5 函数 GPTIM_Set_Period

表 12-8 函数 GPTIM_Set_Period

函数名	GPTIM_Set_Period
函数原型	void GPTIM_Set_Period (GPTIM_SFRmap* GPTIMx, uint32_t Period)
功能描述	更新定时器周期值。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	Period: 新的周期值, 取值 16 位数据。
返回值	无
被调用函数	无

12.4.6 函数 GPTIM_Set_Prescaler

表 12-9 函数 GPTIM_Set_Prescaler

函数名	GPTIM_Set_Prescaler
函数原型	void GPTIM_Set_Prescaler (GPTIM_SFRmap* GPTIMx, uint32_t Prescaler)
功能描述	更新定时器预分频值。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	Prescaler: 新的预分频, 取值 32 位或 16 位数据。
返回值	无
被调用函数	无

12.4.7 函数 GPTIM_Counter_Mode_Config

表 12-10 函数 GPTIM_Counter_Mode_Config

函数名	GPTIM_Counter_Mode_Config
函数原型	void GPTIM_Counter_Mode_Config (GPTIM_SFRmap* GPTIMx, uint32_t CounterMode)
功能描述	更新定时器计数模式。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。

输入参数 2	CounterMode: 新的计数模式, 取值范围为: GPTIM_COUNT_DOWN_UF: 向下计数, 下溢产生中断标志 GPTIM_COUNT_UP_OF: 向上计数, 上溢产生中断标志 GPTIM_COUNT_UP_DOWN_OF: 向上-向下计数, 上溢产生中断标志 GPTIM_COUNT_UP_DOWN_UF: 向上-向下计数, 下溢产生中断标志 GPTIM_COUNT_UP_DOWN_OUF: 向上-向下计数, 上溢和下溢产生中断标志
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

12.4.8 函数 GPTIM_Clock_Config

表 12-11 函数 GPTIM_Clock_Config

函数名	GPTIM_Clock_Config
函数原型	void GPTIM_Clock_Config (GPTIM_SFRmap* GPTIMx, uint32_t NewClock)
功能描述	更新定时器工作时钟。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

12.4.9 函数 GPTIM_External_Pulse_Sync_Config

表 12-12 函数 GPTIM_External_Pulse_Sync_Config

函数名	GPTIM_External_Pulse_Sync_Config
函数原型	void GPTIM_External_Pulse_Sync_Config (GPTIM_SFRmap* GPTIMx, uint32_t PulseSync)
功能描述	更新定时器计数模式外部触发脉冲输入同步控制位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewClock: 新的定时器计数模式外部触发脉冲输入同步控制位使能状态, 取值范围为: GPTIM_EX_SYNC_MODE: 与外部触发脉冲输入同步 GPTIM_NO_SYNC_MODE: 不与外部触发脉冲输入同步
返回值	无
被调用函数	无

12.4.10 函数 GPTIM_Work_Mode_Config

表 12-13 函数 GPTIM_Work_Mode_Config

函数名	GPTIM_Work_Mode_Config
函数原型	void GPTIM_Work_Mode_Config (GPTIM_SFRmap* GPTIMx, uint32_t NewState)
功能描述	更新定时/计数模式选择。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewState: 新的定时/计数模式，取值范围为： GPTIM_TIMER_MODE: 定时模式 GPTIM_COUNTER_MODE: 计数模式
返回值	无
被调用函数	无

12.4.11 函数 GPTIM_Udata_Immediately_Config

表 12-14 函数 GPTIM_Udata_Immediately_Config

函数名	GPTIM_Udata_Immediately_Config
函数原型	void GPTIM_Udata_Immediately_Config (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	配置立即更新控制位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewState: 立即更新使能状态，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

12.4.12 函数 GPTIM_Master_Slave_Snyc_Config

表 12-15 函数 GPTIM_Master_Slave_Snyc_Config

函数名	GPTIM_Master_Slave_Snyc_Config
函数原型	void GPTIM_Master_Slave_Snyc_Config (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	配置主从模式同步位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewState: 主从模式同步位状态，取值范围为：TRUE 或 FALSE

返回值	无
被调用函数	无

12.4.13 函数 GPTIM_Trigger_Select_Config

表 12-16 函数 GPTIM_Trigger_Select_Config

函数名	GPTIM_Trigger_Select_Config
函数原型	void GPTIM_Trigger_Select_Config (GPTIM_SFRmap* GPTIMx, uint32_t TriggerSelect)
功能描述	配置触发选择位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	TriggerSelect: 触发选择，取值范围为： GPTIM_TRIGGER_T1 GPTIM_TRIGGER_T2 GPTIM_TRIGGER_T3 GPTIM_TRIGGER_T4 GPTIM_TRIGGER_T5 GPTIM_TRIGGER_T9 GPTIM_TRIGGER_T14 GPTIM_TRIGGER_T15 GPTIM_TRIGGER_T18 GPTIM_TRIGGER_T19 GPTIM_TRIGGER_T20 GPTIM_TRIGGER_T21 GPTIM_TRIGGER_TXCK GPTIM_TRIGGER_CCPXCH1 GPTIM_TRIGGER_CCPXCH2 GPTIM_TRIGGER_CCPXCH3
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

12.4.14 函数 GPTIM_Slave_Mode_Config

表 12-17 函数 GPTIM_Slave_Mode_Config

函数名	GPTIM_Slave_Mode_Config
函数原型	void GPTIM_Slave_Mode_Config (GPTIM_SFRmap* GPTIMx, uint32_t SlaveMode)
功能描述	配置从模式选择位。

输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	SlaveSelect: 从模式选择, 取值范围为: GPTIM_SLAVE_FORBIDDEN_MODE: 从模式禁止 GPTIM_SLAVE_TRIGGER_MODE: 触发模式 GPTIM_SLAVE_GATED_MODE: 门控模式 GPTIM_SLAVE_RESET_MODE: 复位模式 GPTIM_SLAVE_COUNTER_MODE: 计数模式 2
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

12.4.15 函数 GPTIM_Master_Mode_Config

表 12-18 函数 GPTIM_Master_Mode_Config

函数名	GPTIM_Master_Mode_Config
函数原型	void GPTIM_Master_Mode_Config (GPTIM_SFRmap* GPTIMx, uint32_t MasterMode)
功能描述	配置主模式选择位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	MasterMode: 主模式选择, 取值范围为: GPTIM_MASTER_UR_SIGNAL: UR 位作为触发 GPTIM_MASTER_EN_SIGNAL: TXEN 作为触发 GPTIM_MASTER_TXIF_SIGNAL: TXIF 作为触发 GPTIM_MASTER_CCPXCH1IF_SIGNAL: CCPxCH1IF 脉冲作为触发 GPTIM_MASTER_CCPXCH1_SIGNAL: CCPxCH1 作为触发 GPTIM_MASTER_CCPXCH2_SIGNAL: CCPxCH2 作为触发 GPTIM_MASTER_CCPXCH3_SIGNAL: CCPxCH3 作为触发 GPTIM_MASTER_CCPXCH4_SIGNAL: CCPxCH4 作为触发
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

12.4.16 函数 GPTIM_Udata_Rising_Edge_Config

表 12-19 函数 GPTIM_Udata_Rising_Edge_Config

函数名	GPTIM_Udata_Rising_Edge_Config
函数原型	void GPTIM_Udata_Rising_Edge_Config (GPTIM_SFRmap* GPTIMx,

	FunctionalState NewState)
功能描述	配置上升沿更新事件控制位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewState: 更新事件控制状态, 取值范围为: TRUE: 上升沿立即更新 FALSE: 每周期更新
返回值	无
被调用函数	无

12.4.17 函数 GPTIM_Update_Enable

表 12-20 函数 GPTIM_Update_Enable

函数名	GPTIM_Update_Enable
函数原型	void GPTIM_Update_Enable (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	配置更新使能。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
输入参数 2	NewState: 更新使能状态, 取值范围为: TRUE: 允许更新 FALSE: 禁止更新
返回值	无
被调用函数	无

12.4.18 函数 GPTIM_Trigger_DMA_Enable

表 12-21 函数 GPTIM_Trigger_DMA_Enable

函数名	GPTIM_Trigger_DMA_Enable
函数原型	void GPTIM_Trigger_DMA_Enable (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	允许触发事件的 DMA 请求配置
输入参数 1	CCPx: 指向 CCP 或通用定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR, 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
输入参数 2	NewState: 触发事件的 DMA 请求使能状态, 取值范围为: TRUE 或 FALSE

返回值	无
被调用函数	无

12.4.19 函数 GPTIM_Update_DMA_Enable

表 12-22 函数 GPTIM_Update_DMA_Enable

函数名	GPTIM_Update_DMA_Enable
函数原型	void GPTIM_Update_DMA_Enable (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	配置更新事件的 DMA 请求使能。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR, 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SF R。
返回值	无
被调用函数	无

12.4.20 函数 GPTIM_Update_INT_Enable

表 12-23 函数 GPTIM_Update_INT_Enable

函数名	GPTIM_Update_INT_Enable
函数原型	void GPTIM_Update_INT_Enable (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	配置 Tx 更新事件中断使能。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR, 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SF R。
输入参数 2	NewState: Tx 更新事件中断, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

12.4.21 函数 GPTIM_Trigger_INT_Enable

表 12-24 函数 GPTIM_Trigger_INT_Enable

函数名	GPTIM_Trigger_INT_Enable
-----	--------------------------

函数原型	void GPTIM_Trigger_INT_Enable (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	配置 Tx 触发事件中断使能。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SF R。
输入参数 2	NewState: Tx 触发事件中断，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

12.4.22 函数 GPTIM_Generate_Trigger_Config

表 12-25 函数 GPTIM_Generate_Trigger_Config

函数名	GPTIM_Generate_Trigger_Config
函数原型	void GPTIM_Generate_Trigger_Config (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	产生触发事件配置位。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SF R。
输入参数 2	NewState: 定时器使能控制状态，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

12.4.23 函数 GPTIM_Get_Direction

表 12-26 函数 GPTIM_Get_Direction

函数名	GPTIM_Get_Direction
函数原型	DIRStatus GPTIM_Get_Direction (GPTIM_SFRmap* GPTIMx)
功能描述	读 TX 计数方向。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
返回值	TX 计数方向，0: 向下，1: 向上。

被调用函数	无
-------	---

12.4.24 函数 GPTIM_Get_Counter

表 12-27 函数 GPTIM_Get_Counter

函数名	GPTIM_Get_Counter
函数原型	uint32_t GPTIM_Get_Counter (GPTIM_SFRmap* GPTIMx)
功能描述	读定时器计数值。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
返回值	Tx 计数值，32 位
被调用函数	无

12.4.25 函数 GPTIM_Get_Period

表 12-28 函数 GPTIM_Get_Period

函数名	GPTIM_Get_Period
函数原型	uint32_t GPTIM_Get_Period (GPTIM_SFRmap* GPTIMx)
功能描述	读定时器周期值。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
返回值	Tx 周期值，32 位
被调用函数	无

12.4.26 函数 GPTIM_Get_Prescaler

表 12-29 函数 GPTIM_Get_Prescaler

函数名	GPTIM_Get_Prescaler
函数原型	uint32_t GPTIM_Get_Prescaler (GPTIM_SFRmap* GPTIMx)
功能描述	读定时器预分频值。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
返回值	Tx 预分频值，32 位
被调用函数	无

12. 4. 27 函数 GPTIM_Overflow_INT_Enable

表 12-30 函数 GPTIM_Overflow_INT_Enable

函数名	GPTIM_Overflow_INT_Enable
函数原型	void GPTIM_Overflow_INT_Enable (GPTIM_SFRmap* GPTIMx, FunctionalState NewState)
功能描述	读取 Tx 计数溢出中断使能。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
输入参数 2	NewState: Tx 计数溢出中断，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

12. 4. 28 函数 GPTIM_Clear_Overflow_INT_Flag

表 12-31 函数 GPTIM_Clear_Overflow_INT_Flag

函数名	GPTIM_Clear_Overflow_INT_Flag
函数原型	void GPTIM_Clear_Overflow_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	清除 Tx 溢出中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SF R。
返回值	无
被调用函数	无

12. 4. 29 函数 GPTIM_Clear_Update_INT_Flag

表 12-32 函数 GPTIM_Clear_Update_INT_Flag

函数名	GPTIM_Clear_Update_INT_Flag
函数原型	void GPTIM_Clear_Update_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	清除 Tx 更新事件中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/

	CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
返回值	无
被调用函数	无

12.4.30 函数 GPTIM_Clear_Trigger_INT_Flag

表 12-33 函数 GPTIM_Clear_Trigger_INT_Flag

函数名	GPTIM_Clear_Trigger_INT_Flag
函数原型	void GPTIM_Clear_Trigger_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	清除 Tx 触发事件中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
返回值	无
被调用函数	无

12.4.31 函数 GPTIM_Get_Overflow_INT_Flag

表 12-34 函数 GPTIM_Get_Overflow_INT_Flag

函数名	GPTIM_Get_Overflow_INT_Flag
函数原型	FlagStatus GPTIM_Get_Overflow_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	读取 Tx 计数溢出中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR， 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
返回值	中断状态，0：未发生中断，1：发生中断
被调用函数	无

12.4.32 函数 GPTIM_Get_Update_INT_Flag

表 12-35 函数 GPTIM_Get_Update_INT_Flag

函数名	GPTIM_Get_Update_INT_Flag
函数原型	FlagStatus GPTIM_Get_Update_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	读取 Tx 更新事件中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针， 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/

	T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR, 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SF R。
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

12.4.33 函数 GPTIM_Get_Trigger_INT_Flag

表 12-36 函数 GPTIM_Get_Trigger_INT_Flag

函数名	GPTIM_Get_Trigger_INT_Flag
函数原型	FlagStatus GPTIM_Get_Trigger_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	读取 Tx 触发事件中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR。
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

12.4.34 函数 GPTIM_Get_Updata_DMA_INT_Flag

表 12-37 函数 GPTIM_Get_Updata_DMA_INT_Flag

函数名	GPTIM_Get_Updata_DMA_INT_Flag
函数原型	FlagStatus GPTIM_Get_Updata_DMA_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	读取 Tx 更新事件触发 DMA 中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针, 取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR, 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
返回值	中断状态, 0: 未发生中断, 1: 发生中断, DMA 响应后该位由硬件自动清零
被调用函数	无

12.4.35 函数 GPTIM_Get_Trigger_DMA_INT_Flag

表 12-38 函数 GPTIM_Get_Trigger_DMA_INT_Flag

函数名	GPTIM_Get_Trigger_DMA_INT_Flag
函数原型	FlagStatus GPTIM_Get_Trigger_DMA_INT_Flag (GPTIM_SFRmap* GPTIMx)
功能描述	配置触发事件触发 DMA 中断标志。
输入参数 1	GPTIMx: 指向定时器内存结构的指针,

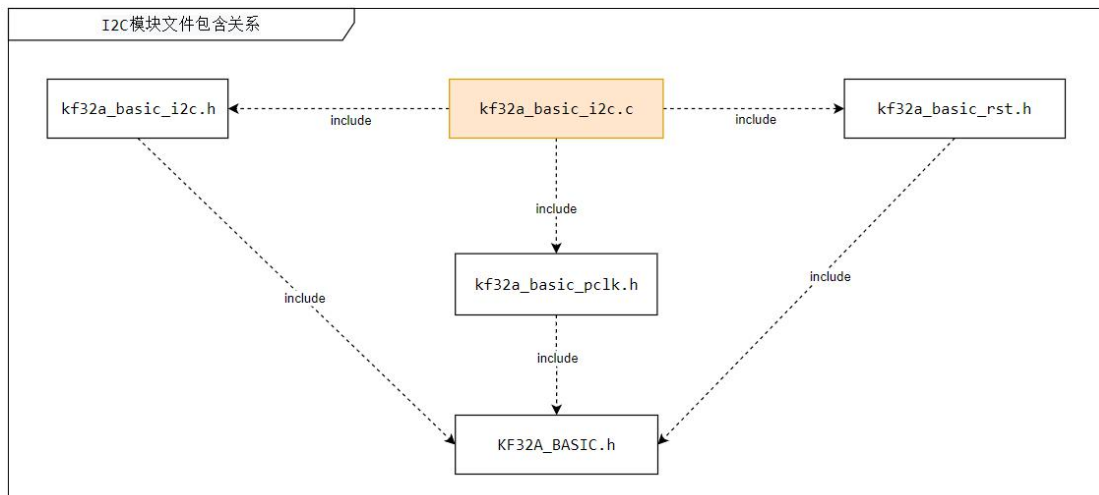
	取值 T0_SFR/T1_SFR/T2_SFR/T3_SFR/T4_SFR/T18_SFR/ T19_SFR/T20_SFR/T21_SFR/T19_SFR/T22_SFR/T23_SFR, 或 CCP0_SFR/CCP1_SFR/CCP2_SFR/CCP3_SFR/CCP4_SFR/ CCP18_SFR/CCP19_SFR/CCP20_SFR/CCP21_SFR/CCP22_SFR/CCP23_SFR。
返回值	中断状态，0：未发生中断，1：发生中断，DMA 响应后该位由硬件自动清零
被调用函数	无

13 内部集成电路接口（I2C）

KungFu32 内核提供了多个独立的 I2C（Inter—Integrated Circuit）。I2C 模块能实现全部从动功能，且硬件支持启动位和停止位中断，以便于固件实现主控功能。I2C 模块实现标准模式规范以及 7 位和 10 位寻址。有两个引脚用于数据传输：时钟线（SCL）和数据线（SDA）。

13.1 文件引用关系

图 13-1 I2C 寄存器结构说明



13.2 I2C 寄存器结构

I2C 寄存器结构，I2C_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct I2C_MemMap {
    volatile uint32_t    CTLR;
    volatile uint32_t    SR;
    volatile uint32_t    BUFR;
    volatile uint32_t    ADDR0;
    volatile uint32_t    BRGR;
    volatile uint32_t    ADDR1;
    volatile uint32_t    ADDR2;
    volatile uint32_t    ADDR3;
    volatile uint32_t    IER;
    volatile uint32_t    TPSR;
}I2C_SFRmap;
    
```

表 13-1 I2C 寄存器结构说明

寄存器	描述
CTLR	I2C 控制寄存器,偏移:0x0

SR	I2C 状态寄存器,偏移:0x4
BUFR	I2C 数据寄存器,偏移:0x8
ADDR0	I2C 地址寄存器 0,偏移:0xC
BRGR	I2C 波特率寄存器,偏移:0x10
ADDR1	I2C 地址寄存器 1,偏移:0x14
ADDR2	I2C 地址寄存器 2,偏移:0x18
ADDR3	I2C 地址寄存器 3,偏移:0x1C
IER	I2C 中断使能寄存器,偏移:0x20
TPSR	I2C 测试引脚选择寄存器,偏移:0x24

I2C 配置信息结构体, I2C_InitTypeDef, 定义于文件 kf32a_basic_i2c.h 中, 如下:

```
typedef struct
{
    uint32_t    m_Mode;
    uint32_t    m_ClockSource;
    uint32_t    m_BADR10;
    uint32_t    m_MasterSlave;
    uint16_t    m_BaudRateL;
    uint16_t    m_BaudRateH;
    FunctionalState    m_AckEn;
    uint32_t    m_AckData;
}I2C_InitTypeDef;
```

表 13-2 I2C 配置信息结构体说明

配置信息	描述
m_Mode	I2C 模式选择配置, 取值为宏 “I2C 模式” 中的一个。
m_ClockSource	I2C 工作时钟, 取值为宏 “I2C 工作时钟” 中的一个。
m_BADR10	I2C 地址选择配置, 取值为宏 “I2C 地址选择” 中的一个。
m_MasterSlave	SMBus 类型选择, 取值为宏 “SMBus 类型选择” 中的一个。
m_BaudRateL	SCL 低电平占用的时钟周期数, 取值为 0~0xFFFF。
m_BaudRateH	SCL 高电平占用的时钟周期数, 取值为 0~0xFFFF。
m_AckEn	I2C 应答使能, 取值为 TRUE 或 FALSE。
m_AckData	应答数据位, 取值为宏 “I2C 应答数据位” 中的一个。

13.3 I2C 宏定义

I2C 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, I2C 其他相关宏定义详见文件 kf32a_basic_i2c.h 及函数参数描述。

13.4 I2C 库函数

表 13-3 I2C 固件库函数列表

序号	函数名	描述
1	I2C_Reset	I2C 外设复位。
2	I2C_Configuration	I2C 外设配置。
3	I2C_Struct_Init	初始化 I2C 配置信息结构体。
4	I2C_Cmd	控制 I2C 使能位。
5	I2C_Bufr_Address_Config	控制 I2C10 位地址使能位。
6	I2C_Generate_START	控制 I2C_启动使能位。
7	I2C_Generate_STOP	控制 I2C 停止条件使能位。
8	I2C_Ack_Config	控制 I2C_Ack 应答使能位。
9	I2C_Ack_DATA_Config	控制 I2C_Ack 应答数据位。
10	I2C_Call_Cmd	控制 I2C_Call 使能位。
11	I2C_Clock_Config	配置 I2C 工作时钟选择寄存器。
12	I2C_MATCH_ADDRESS_Config	控制 I2C 地址寄存器匹配位。
13	I2C_SCL_Enable	控制 I2C SCL 输出使能。
14	I2C_NMENA_Enable	控制 I2C 监控模式使能位。
15	I2C_SMBUS_Enable	控制 SMBus 模式使能信号。
16	I2C_SMBT_Config	控制 SMBUS 类型。
17	I2C_SMBus_ALERT_Config	控制 SMBus 提醒。
18	I2C_SendData	I2C 发送数据。
19	I2C_SendData8	I2C 发送 8 位数据。
20	I2C_ReceiveData	I2C 接收数据。
21	I2C_ARP_Enable	控制 I2C_ARP 使能位。
22	I2C_ADDR_Config	配置 I2C 地址位。
23	I2C_MSK_Config	配置 I2C 地址屏蔽位。
24	I2C_BRGH_Config	配置 I2C SCL 高电平占用的时钟周期数。
25	I2C_BRGL_Config	配置 I2C SCL 低电平占用的时钟周期数。
26	I2C_Start_INT_Enable	设置 I2C 起始信号中断使能。
27	I2C_Stop_INT_Enable	设置 I2C 停止信号中断使能。
28	I2C_Ack_Fail_INT_Enable	设置 I2C 应答错误中断使能。
29	I2C_Arbitration_Lost_INT_Enable	设置 I2C 失去仲裁中断使能。
30	I2C_SMBus_Alert_INT_Enable	设置 SMBus 提醒中断使能。
31	I2C_SMBus_HostHead_INT_Enable	设置 SMBus 主机头系列中断使能。
32	I2C_SMBus_Device_Defaultaddress_INT_Enable	设置 SMBus 设备默认地址中断使能。
33	I2C_ISIE_INT_Enable	设置 I2C 中断信号使能。

34	I2C_Receive_DMA_INT_Enable	设置 I2C 接收 DMA 中断使能。
35	I2C_Transmit_DMA_INT_Enable	设置 I2C 发送 DMA 中断使能。
36	I2C_Get_Start_Flag	获取 I2C 起始信号标志位状态
37	I2C_Clear_Start_Flag	清零 I2C 起始信号标志。
38	I2C_Get_Stop_Flag	获取 I2C 停止信号标志位状态
39	I2C_Clear_Stop_Flag	清零 I2C 停止信号标志。
40	I2C_Get_Address_Match_Flag	获取 I2C 地址匹配状态位状态
41	I2C_Get_HighAddress_Flag	获取 I2C 高位地址状态位状态
42	I2C_Get_Data_Flag	获取 I2C 数据内容状态位状态
43	I2C_Get_Ack_Fail_Flag	获取 I2C 应答错误标志位状态。
44	I2C_Clear_Ack_Fail_Flag	清零 I2C 应答错误标志。
45	I2C_Get_Arbitration_Lost_Flag	获取 I2C 失去仲裁标志位状态
46	I2C_Get_Write_Read_Flag	获取 I2C 读/ 写信息状态位状态。
47	I2C_Get_SMBus_Alert_Flag	获取 SMBus 提醒中断标志状态。
48	I2C_Clear_SMBus_Alert_Flag	清零 SMBus 提醒中断标志。
49	I2C_Get_SMBus_Host_Header_Flag	获取 SMBus 主机头系列中断标志状态
50	I2C_Clear_SMBus_Host_Header_Flag	清零 SMBus 主机头系列中断标志。
51	I2C_Get_SMBus_Device_Default_Flag	获取 SMBus 设备默认地址(从模式)。
52	I2C_Clear_SMBus_Device_Default_Flag	清零 SMBus 设备默认地址(从模式)标志。
53	I2C_Get_INTERRUPT_Flag	获取 I2C 中断信号标志位状态
54	I2C_Clear_INTERRUPT_Flag	清零 I2C 中断信号标志。
55	I2C_Get_Receive_Buff_Flag	获取 I2C 接收 BUFF 为满状态
56	I2C_Get_Transmit_Buff_Flag	获取 I2C 发送 BUFF 状态位
57	I2C_Get_Receive_DMA_Flag	获取 I2C 接收 DMA 中断标志位
58	I2C_Get_Transmit_DMA_Flag	获取 I2C 发送 DMA 中断标志位

13.4.1 函数 I2C_Reset

表 13-4 函数 I2C_Reset

函数名	I2C_Reset
函数原型	void I2C_Reset (I2C_SFRmap* I2Cx)
功能描述	I2C 外设复位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
返回值	无
被调用函数 1	RST_CTL1_Peripheral_Reset_Enable
被调用函数 2	PCLK_CTL1_Peripheral_Clock_Enable
被调用函数 3	RST_CTL3_Peripheral_Reset_Enable

被调用函数 4	PCLK_CTL3_Peripheral_Clock_Enable
---------	-----------------------------------

13.4.2 函数 I2C_Configuration

表 13-5 函数 I2C_Configuration

函数名	I2C_Configuration
函数原型	void I2C_Configuration (I2C_SFRmap* I2Cx, I2C_InitTypeDef* i2cInitStruct)
功能描述	I2C 外设配置。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	i2cInitStruct: I2C 模块配置信息结构体指针。
返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

13.4.3 函数 I2C_Struct_Init

表 13-6 函数 I2C_Struct_Init

函数名	I2C_Struct_Init
函数原型	void I2C_Struct_Init (I2C_InitTypeDef* I2C_InitStruct)
功能描述	初始化 I2C 配置信息结构体。
输入参数 1	dacInitStruct: 指向待初始化的结构体指针。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.4 函数 I2C_Cmd

表 13-7 函数 I2C_Cmd

函数名	I2C_Cmd
函数原型	void I2C_Cmd(I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C 使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 使能位配置信息, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.5 函数 I2C_Bufr_Address_Config

表 13-8 函数 I2C_Bufr_Address_Config

函数名	I2C_Bufr_Address_Config
函数原型	void I2C_Bufr_Address_Config(I2C_SFRmap* I2Cx, uint32_t NewState)
功能描述	控制 I2C10 位地址使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 使能位配置信息, 取值为 I2C_BUFRADDRESS_10BIT 或 I2C_BUFRADDRESS_7BIT。
返回值	无
被调用函数 1	无

13.4.6 函数 I2C_Generate_START

表 13-9 函数 I2C_Generate_START

函数名	I2C_Generate_START
函数原型	无
功能描述	控制 I2C_启动使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 启动使能位配置信息, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.7 函数 I2C_Generate_STOP

表 13-10 函数 I2C_Generate_STOP

函数名	I2C_Generate_STOP
函数原型	void I2C_Generate_STOP(I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C 停止条件使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 停止条件使能位配置信息, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.8 函数 I2C_Ack_Config

表 13-11 函数 I2C_Ack_Config

函数名	I2C_Ack_Config
函数原型	void I2C_Ack_Config (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C_Ack 应答使能位。

输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 应答使能位配置信息, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.9 函数 I2C_Ack_DATA_Config

表 13-12 函数 I2C_Ack_DATA_Config

函数名	I2C_Ack_DATA_Config
函数原型	void I2C_Ack_DATA_Config (I2C_SFRmap* I2Cx, uint32_t NewState)
功能描述	控制 I2C_Ack 应答数据位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 应答数据位配置信息, 取值为 I2C_ACKDATA_ACK: 应答 I2C_ACKDATA_NO_ACK: 不应答
返回值	无
被调用函数 1	无

13.4.10 函数 I2C_Call_Cmd

表 13-13 函数 I2C_Call_Cmd

函数名	I2C_Call_Cmd
函数原型	void I2C_Call_Cmd (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C_Call 使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C_Call 使能位配置信息, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.11 函数 I2C_Clock_Config

表 13-14 函数 I2C_Clock_Config

函数名	I2C_Clock_Config
函数原型	void I2C_Clock_Config (I2C_SFRmap* I2Cx, uint32_t ClkSource)
功能描述	配置 I2C 工作时钟选择寄存器。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	输入 ClkSource: 时钟选择, 取值范围为: I2C_CLK_SCLK: 选用 SCLK 为 I2C 工作时钟 I2C_CLK_HFCLK: 选用 HFCLK 为 I2C 工作时钟 I2C_CLK_LFCLK: 选用 LFCLK 为 I2C 工作时钟

返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

13.4.12 函数 I2C_MATCH_ADDRESS_Config

表 13-15 函数 I2C_MATCH_ADDRESS_Config

函数名	I2C_MATCH_ADDRESS_Config
函数原型	void I2C_MATCH_ADDRESS_Config (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C 地址寄存器匹配位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C_地址寄存器匹配位, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.13 函数 I2C_SCL_Enable

表 13-16 函数 I2C_SCL_Enable

函数名	I2C_SCL_Enable
函数原型	void I2C_SCL_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C_SCL 输出使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C_SCL 输出使能, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.14 函数 I2C_NMENA_Enable

表 13-17 函数 I2C_NMENA_Enable

函数名	I2C_NMENA_Enable
函数原型	void I2C_NMENA_Enable(I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C 监控模式使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C_监控模式使能位, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.15 函数 I2C_SMBUS_Enable

表 13-18 函数 I2C_SMBUS_Enable

函数名	I2C_SMBUS_Enable
函数原型	void I2C_SMBUS_Enable(I2C_SFRmap* I2Cx, uint32_t NewState)
功能描述	控制 SMBus 模式使能信号。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: SMBus 模式使能信号, 取值为: I2C_MODE_SMBUS: SMBus 模式 I2C_MODE_I2C: I2C 模式
返回值	无
被调用函数 1	无

13.4.16 函数 I2C_SMBT_Config

表 13-19 函数 I2C_SMBT_Config

函数名	I2C_SMBT_Config
函数原型	void I2C_SMBT_Config(I2C_SFRmap* I2Cx, uint32_t NewState)
功能描述	控制 SMBUS 类型。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: SMBUS 类型, 取值为: I2C_MODE_SMBUSHOST: SMBus 主机 I2C_MODE_SMBUSDEVICE: SMBus 设备
返回值	无
被调用函数 1	无

13.4.17 函数 I2C_SMBus_ALERT_Config

表 13-20 函数 I2C_SMBus_ALERT_Config

函数名	I2C_SMBus_ALERT_Config
函数原型	void I2C_SMBus_ALERT_Config(I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 SMBus 提醒。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: SMBus 提醒, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.18 函数 I2C_SendData

表 13-21 函数 I2C_SendData

函数名	I2C_SendData
函数原型	void I2C_SendData (I2C_SFRmap* I2Cx, uint32_t Data)
功能描述	I2C 发送数据。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	Data: 写入数据寄存器的值, 取值为 10 位数据。
返回值	无
被调用函数 1	无

13.4.19 函数 I2C_SendData8

表 13-22 函数 I2C_SendData8

函数名	I2C_SendData8
函数原型	void I2C_SendData8 (I2C_SFRmap* I2Cx, uint8_t Data)
功能描述	I2C 发送 8 位数据。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	Data: 写入数据寄存器的值, 取值为 8 位数据。
返回值	无
被调用函数 1	无

13.4.20 函数 I2C_ReceiveData

表 13-23 函数 I2C_ReceiveData

函数名	I2C_ReceiveData
函数原型	uint32_t I2C_ReceiveData(I2C_SFRmap* I2Cx)
功能描述	I2C 接收数据。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.21 函数 I2C_ARP_Enable

表 13-24 函数 I2C_ARP_Enable

函数名	I2C_ARP_Enable
函数原型	void I2C_ARP_Enable(I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	控制 I2C_ARP 使能位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。

输入参数 2	NewState: I2C_ARP 使能位配置信息，取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.22 函数 I2C_ADDR_Config

表 13-25 函数 I2C_ADDR_Config

函数名	I2C_ADDR_Config
函数原型	无
功能描述	配置 I2C 地址位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针，取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	AddrSelect: 地址寄存器选择，取值为 0x0~0x3。
输入参数 3	Data: 地址位选择，取值为 0x0~0x3FF。
返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

13.4.23 函数 I2C_MSK_Config

表 13-26 函数 I2C_MSK_Config

函数名	I2C_MSK_Config
函数原型	void I2C_MSK_Config(I2C_SFRmap* I2Cx, uint32_t AddrSelect, uint32_t DataMask)
功能描述	配置 I2C 地址屏蔽位。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针，取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	AddrSelect: 地址寄存器选择，取值为 0x0~0x3。
输入参数 3	DataMask: 地址位选择，取值为 0x0~0x3FF。
返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

13.4.24 函数 I2C_BRGH_Config

表 13-27 函数 I2C_BRGH_Config

函数名	I2C_BRGH_Config
函数原型	void I2C_BRGH_Config (I2C_SFRmap* I2Cx, uint16_t Period)
功能描述	配置 I2C SCL 高电平占用的时钟周期数。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针，取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	Period: SCL 高电平占用的时钟周期数，取值为 0x0~0xFFFF。
返回值	无

被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)
---------	---

13.4.25 函数 I2C_BRGL_Config

表 13-28 函数 I2C_BRGL_Config

函数名	I2C_BRGL_Config
函数原型	void I2C_BRGL_Config (I2C_SFRmap* I2Cx, uint16_t Period)
功能描述	配置 I2C SCL 低电平占用的时钟周期数。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	Period: SCL 低电平占用的时钟周期数, 取值为 0x0~0xFFFF。
返回值	无
被调用函数 1	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

13.4.26 函数 I2C_Start_INT_Enable

表 13-29 函数 I2C_Start_INT_Enable

函数名	I2C_Start_INT_Enable
函数原型	void I2C_Start_INT_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	设置 I2C 起始信号中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 起始信号中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.27 函数 I2C_Stop_INT_Enable

表 13-30 函数 I2C_Stop_INT_Enable

函数名	I2C_Stop_INT_Enable
函数原型	void I2C_Stop_INT_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	设置 I2C 停止信号中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 停止信号中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.28 函数 I2C_Ack_Fail_INT_Enable

表 13-31 函数 I2C_Ack_Fail_INT_Enable

函数名	I2C_Ack_Fail_INT_Enable
函数原型	void I2C_Ack_Fail_INT_Enable (I2C_SFRmap* I2Cx,FunctionalState NewState)
功能描述	设置 I2C 应答错误中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 应答错误中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.29 函数 I2C_Arbitration_Lost_INT_Enable

表 13-32 函数 I2C_Arbitration_Lost_INT_Enable

函数名	I2C_Arbitration_Lost_INT_Enable
函数原型	void I2C_Arbitration_Lost_INT_Enable (I2C_SFRmap* I2Cx,FunctionalState NewState)
功能描述	设置 I2C 失去仲裁中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 失去仲裁中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.30 函数 I2C_SMBus_Alert_INT_Enable

表 13-33 函数 I2C_SMBus_Alert_INT_Enable

函数名	I2C_SMBus_Alert_INT_Enable
函数原型	void I2C_SMBus_Alert_INT_Enable (I2C_SFRmap* I2Cx,FunctionalState NewState)
功能描述	设置 SMBus 提醒中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: SMBus 提醒中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.31 函数 I2C_SMBus_HostHead_INT_Enable

表 13-34 函数 I2C_SMBus_HostHead_INT_Enable

函数名	I2C_SMBus_HostHead_INT_Enable
-----	-------------------------------

函数原型	void I2C_SMBus_HostHead_INT_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	设置 SMBus 主机头系列中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: SMBus 主机头系列中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.32 函数 I2C_SMBus_Device_Defaultaddress_INT_Enable

表 13-35 函数 I2C_SMBus_Device_Defaultaddress_INT_Enable

函数名	I2C_SMBus_Device_Defaultaddress_INT_Enable
函数原型	void I2C_SMBus_Device_Defaultaddress_INT_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	设置 SMBus 设备默认地址中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: SMBus 设备默认地址中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.33 函数 I2C_ISIE_INT_Enable

表 13-36 函数 I2C_ISIE_INT_Enable

函数名	I2C_ISIE_INT_Enable
函数原型	void I2C_ISIE_INT_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	设置 I2C 中断信号使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 中断信号状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.34 函数 I2C_Receive_DMA_INT_Enable

表 13-37 函数 I2C_Receive_DMA_INT_Enable

函数名	I2C_Receive_DMA_INT_Enable
函数原型	void I2C_Receive_DMA_INT_Enable (I2C_SFRmap* I2Cx, FunctionalState NewState)
功能描述	设置 I2C 接收 DMA 中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 接收 DMA 中断使能状态, 取值为 TRUE 或 FALSE。

返回值	无
被调用函数 1	无

13.4.35 函数 I2C_Transmit_DMA_INT_Enable

表 13-38 函数 I2C_Transmit_DMA_INT_Enable

函数名	I2C_Transmit_DMA_INT_Enable
函数原型	void I2C_Transmit_DMA_INT_Enable (I2C_SFRmap* I2Cx,FunctionalState NewState)
功能描述	设置 I2C 发送 DMA 中断使能。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	NewState: I2C 发送 DMA 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数 1	无

13.4.36 函数 I2C_Get_Start_Flag

表 13-39 函数 I2C_Get_Start_Flag

函数名	I2C_Get_Start_Flag
函数原型	FlagStatus I2C_Get_Start_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 起始信号标志位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 总线上出现了起始位; 0: 总线上未出现了起始位。
被调用函数 1	无

13.4.37 函数 I2C_Clear_Start_Flag

表 13-40 函数 I2C_Clear_Start_Flag

函数名	I2C_Clear_Start_Flag
函数原型	void I2C_Clear_Start_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 I2C 起始信号标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.38 函数 I2C_Get_Stop_Flag

表 13-41 函数 I2C_Get_Stop_Flag

函数名	I2C_Get_Stop_Flag
函数原型	FlagStatus I2C_Get_Stop_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 停止信号标志位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 总线上出现了停止位; 0: 总线上未出现了停止位。
被调用函数 1	无

13.4.39 函数 I2C_Clear_Stop_Flag

表 13-42 函数 I2C_Clear_Stop_Flag

函数名	I2C_Clear_Stop_Flag
函数原型	void I2C_Clear_Stop_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 I2C 停止信号标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.40 函数 I2C_Get_Address_Match_Flag

表 13-43 函数 I2C_Get_Address_Match_Flag

函数名	I2C_Get_Address_Match_Flag
函数原型	FlagStatus I2C_Get_Address_Match_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 地址匹配状态位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 从机收到匹配地址且应答; (总线上出现停止位将硬件清零该状态位) 0: 从机未收到对应地址。
被调用函数 1	无

13.4.41 函数 I2C_Get_HighAddress_Flag

表 13-44 函数 I2C_Get_HighAddress_Flag

函数名	I2C_Get_HighAddress_Flag
-----	--------------------------

函数原型	FlagStatus I2C_Get_HighAddress_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 高位地址状态位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 上次接收或发送的字节是高位地址; 0: 上次接收或发送的字节不是高位地址。
被调用函数 1	无

13.4.42 函数 I2C_Get_Data_Flag

表 13-45 函数 I2C_Get_Data_Flag

函数名	I2C_Get_Data_Flag
函数原型	FlagStatus I2C_Get_Data_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 数据内容状态位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 表示上次接收或发送的字节是数据 (总线上出现起始位将清零该状态位) 0: 表示上次接收或发送的字节是地址。
被调用函数 1	无

13.4.43 函数 I2C_Get_Ack_Fail_Flag

表 13-46 函数 I2C_Get_Ack_Fail_Flag

函数名	I2C_Get_Ack_Fail_Flag
函数原型	FlagStatus I2C_Get_Ack_Fail_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 应答错误标志位状态。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 发生了应答错误; 0: 未发生应答错误。
被调用函数 1	无

13.4.44 函数 I2C_Clear_Ack_Fail_Flag

表 13-47 函数 I2C_Clear_Ack_Fail_Flag

函数名	I2C_Clear_Ack_Fail_Flag
函数原型	void I2C_Clear_Ack_Fail_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 I2C 应答错误标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。

输入参数 2	无
返回值	无
被调用函数 1	无

13.4.45 函数 I2C_Get_Arbitration_Lost_Flag

表 13-48 函数 I2C_Get_Arbitration_Lost_Flag

函数名	I2C_Get_Arbitration_Lost_Flag
函数原型	FlagStatus I2C_Get_Arbitration_Lost_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 失去仲裁标志位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 发送数据过程中失去仲裁; 0: 发送数据过程中未失去仲裁。
被调用函数 1	无

13.4.46 函数 I2C_Get_Write_Read_Flag

表 13-49 函数 I2C_Get_Write_Read_Flag

函数名	I2C_Get_Write_Read_Flag
函数原型	FlagStatus I2C_Get_Write_Read_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 读/ 写信息状态位状态。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.47 函数 I2C_Get_SMBus_Alert_Flag

表 13-50 函数 I2C_Get_SMBus_Alert_Flag

函数名	I2C_Get_SMBus_Alert_Flag
函数原型	FlagStatus I2C_Get_SMBus_Alert_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 SMBus 提醒中断标志状态。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 读信息状态位; 0: 写信息状态位。
被调用函数 1	无

13.4.48 函数 I2C_Clear_SMBus_Alert_Flag

表 13-51 函数 I2C_Clear_SMBus_Alert_Flag

函数名	I2C_Clear_SMBus_Alert_Flag
函数原型	void I2C_Clear_SMBus_Alert_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 SMBus 提醒中断标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.49 函数 I2C_Get_SMBus_Host_Header_Flag

表 13-52 函数 I2C_Get_SMBus_Host_Header_Flag

函数名	I2C_Get_SMBus_Host_Header_Flag
函数原型	FlagStatus I2C_Get_SMBus_Host_Header_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 SMBus 主机头系列中断标志状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 读信息状态位; 0: 写信息状态位。
被调用函数 1	无

13.4.50 函数 I2C_Clear_SMBus_Host_Header_Flag

表 13-53 函数 I2C_Clear_SMBus_Host_Header_Flag

函数名	I2C_Clear_SMBus_Host_Header_Flag
函数原型	void I2C_Clear_SMBus_Host_Header_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 SMBus 主机头系列中断标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.51 函数 I2C_Get_SMBus_Device_Default_Flag

表 13-54 函数 I2C_Get_SMBus_Device_Default_Flag

函数名	I2C_Get_SMBus_Device_Default_Flag
函数原型	FlagStatus I2C_Get_SMBus_Device_Default_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 SMBus 设备默认地址(从模式)。

输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 读信息状态位; 0: 写信息状态位。
被调用函数 1	无

13.4.52 函数 I2C_Clear_SMBus_Device_Default_Flag

表 13-55 函数 I2C_Clear_SMBus_Device_Default_Flag

函数名	I2C_Clear_SMBus_Device_Default_Flag
函数原型	void I2C_Clear_SMBus_Device_Default_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 SMBus 设备默认地址(从模式)标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.53 函数 I2C_Get_INTERRUPT_Flag

表 13-56 函数 I2C_Get_INTERRUPT_Flag

函数名	I2C_Get_INTERRUPT_Flag
函数原型	FlagStatus I2C_Get_INTERRUPT_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 中断信号标志位状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 产生了 I2C 中断信号标志位; 0: 未产生 I2C 中断信号标志位。
被调用函数 1	无

13.4.54 函数 I2C_Clear_INTERRUPT_Flag

表 13-57 函数 I2C_Clear_INTERRUPT_Flag

函数名	I2C_Clear_INTERRUPT_Flag
函数原型	void I2C_Clear_INTERRUPT_Flag (I2C_SFRmap* I2Cx)
功能描述	清零 I2C 中断信号标志。
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	无
被调用函数 1	无

13.4.55 函数 I2C_Get_Receive_Buff_Flag

表 13-58 函数 I2C_Get_Receive_Buff_Flag

函数名	I2C_Get_Receive_Buff_Flag
函数原型	FlagStatus I2C_Get_Receive_Buff_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 接收 BUFF 为满状态
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 接收 BUFF 为满; (读 I2Cx_BUFR 清零该位) 0: 接收 BUFF 为空。
被调用函数 1	无

13.4.56 函数 I2C_Get_Transmit_Buff_Flag

表 13-59 函数 I2C_Get_Transmit_Buff_Flag

函数名	I2C_Get_Transmit_Buff_Flag
函数原型	FlagStatus I2C_Get_Transmit_Buff_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 发送 BUFF 状态位
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 等待写 I2Cx_BUFR; (写 I2Cx_BUFR 清零该位) 0: 不需要写 I2Cx_BUFR。
被调用函数 1	无

13.4.57 函数 I2C_Get_Receive_DMA_Flag

表 13-60 函数 I2C_Get_Receive_DMA_Flag

函数名	I2C_Get_Receive_DMA_Flag
函数原型	FlagStatus I2C_Get_Receive_DMA_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 接收 DMA 中断标志位
输入参数 1	I2Cx: 指向 I2C 内存结构的指针, 取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 产生了 I2C 接收 DMA 中断; 0: 未产生 I2C 接收 DMA 中断。
被调用函数 1	无

13.4.58 函数 I2C_Get_Transmit_DMA_Flag

表 13-61 函数 I2C_Get_Transmit_DMA_Flag

函数名	I2C_Get_Transmit_DMA_Flag
-----	---------------------------

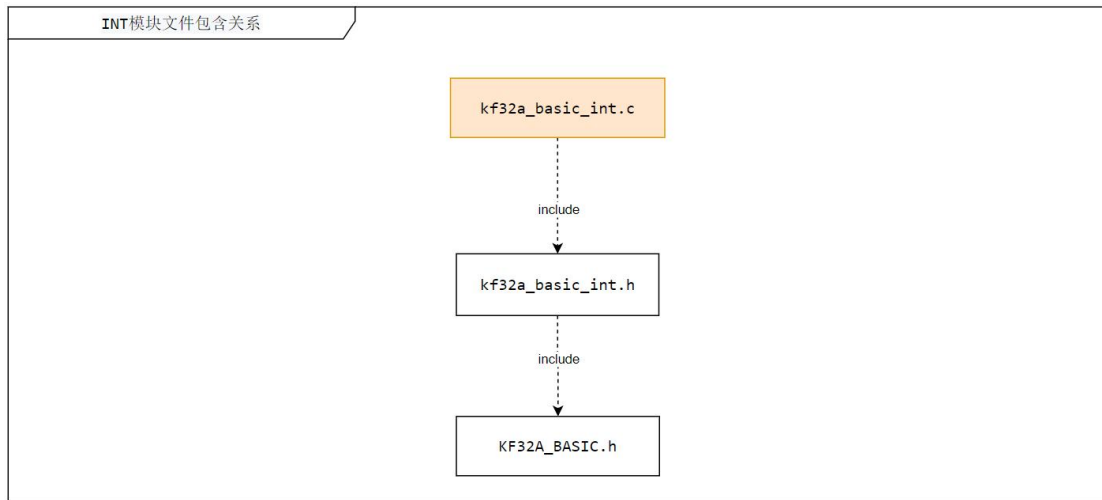
函数原型	FlagStatus I2C_Get_Transmit_DMA_Flag (I2C_SFRmap* I2Cx)
功能描述	获取 I2C 发送 DMA 中断标志位
输入参数 1	I2Cx: 指向 I2C 内存结构的指针，取值为 I2C0_SFR~I2C3_SFR。
输入参数 2	无
返回值	返回 1: 产生了 I2C 发送 DMA 中断； 0: 未产生 I2C 发送 DMA 中断。
被调用函数 1	无

14 中断（INT）

中断模块，支持最多 13 个系统中断，最多 64 个外设中断（含软件中断），16 级中断优先级设置，自动堆栈，中断入口配置，嵌套中断，软件中断。

14.1 文件引用关系

图 14-1 INT 模块文件包含关系



14.2 INT 寄存器结构

INT 寄存器结构，INT_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct INT_MemMap
{
    volatile uint32_t    CTL0;
    volatile uint32_t    EIE0;
    uint32_t    RESERVED1;
    volatile uint32_t    EIE1;
    uint32_t    RESERVED2;
    Volatile uint32_t    EIE2;
    uint32_t    RESERVED3;
    volatile    uint32_t    EIF0;
    uint32_t RESERVED4;
    volatile uint32_t    EIF1;
    uint32_t    RESERVED5;
    volatile uint32_t    EIF2;
    uint32_t    RESERVED6;
}
  
```

```
volatile uint32_t IP0;
volatile uint32_t IP1;
volatile uint32_t IP2;
Volatile uint32_t IP3;
Volatile uint32_t IP4;
volatile uint32_t IP5;
volatile uint32_t IP6;
volatile uint32_t IP7;
Volatile uint32_t IP8;
volatile uint32_t IP9;
volatile uint32_t IP10;
volatile uint32_t IP11;
Volatile uint32_t IP12;
volatile uint32_t IP13;
volatile uint32_t IP14;
volatile uint32_t IP15;
volatile uint32_t IP16;
volatile uint32_t IP17;
volatile uint32_t IP18;
volatile uint32_t EINTMASK;
volatile uint32_t EINTRISE;
volatile uint32_t EINTFALL;
volatile uint32_t EINTF;
uint32_t RESERVED7;
Volatile uint32_t EINTSS0;
volatile uint32_t EINTSS1;
volatile uint32_t CTL1;
} INT_SFRmap;
```

表 14-1 INT 寄存器结构说明

寄存器	描述
CTL0	中断控制寄存器 0, 偏移:0x0000
EIE0	中断使能寄存器 0, 偏移:0x0004
RESERVED1	保留地址, 偏移:0x0008
EIE1	中断使能寄存器 1, 偏移:0x000C
RESERVED2	保留地址, 偏移:0x0010
EIE2	中断使能寄存器 2, 偏移:0x0014
RESERVED3	保留地址, 偏移:0x0018
EIF0	中断标志位寄存器 0, 偏移:0x001C
RESERVED4	保留地址, 偏移:0x0020
EIF1	中断标志位寄存器 1, 偏移:0x0024
RESERVED5	保留地址, 偏移:0x0028
EIF2	中断标志位寄存器 2, 偏移:0x002C

RESERVED6	保留地址, 偏移:0x0030
IP0	中断优先级控制寄存器 0, 偏移:0x0034
IP1	中断优先级控制寄存器 1, 偏移:0x0038
IP2	中断优先级控制寄存器 2, 偏移:0x003C
IP3	中断优先级控制寄存器 3, 偏移:0x0040
IP4	中断优先级控制寄存器 4, 偏移:0x0044
IP5	中断优先级控制寄存器 5, 偏移:0x0048
IP6	中断优先级控制寄存器 6, 偏移:0x004C
IP7	中断优先级控制寄存器 7, 偏移:0x0050
IP8	中断优先级控制寄存器 8, 偏移:0x0054
IP9	中断优先级控制寄存器 9, 偏移:0x0058
IP10	中断优先级控制寄存器 10, 偏移:0x005C
IP11	中断优先级控制寄存器 11, 偏移:0x0060
IP12	中断优先级控制寄存器 12, 偏移:0x0064
IP13	中断优先级控制寄存器 13, 偏移:0x0068
IP14	中断优先级控制寄存器 14, 偏移:0x006C
IP15	中断优先级控制寄存器 15, 偏移:0x0070
IP16	中断优先级控制寄存器 16, 偏移:0x0074
IP17	中断优先级控制寄存器 17, 偏移:0x0078
IP18	中断优先级控制寄存器 18, 偏移:0x007C
EINTMASK	外部中断屏蔽寄存器, 偏移:0x0080
EINTRISE	外部中断上升沿选择寄存器, 偏移:0x0084
EINTFALL	外部中断下降沿选择寄存器, 偏移:0x0088
EINTF	外部中断标志位寄存器, 偏移:0x008C
RESERVED7	保留地址, 偏移:0x0090
EINTSS0	外部中断线中断源选择寄存器 0, 偏移:0x0094
EINTSS1	外部中断线中断源选择寄存器 1, 偏移:0x0098
CTL1	中断控制寄存器 1, 偏移:0x009C

INT 配置信息结构体, 定义于文件 kf32a_basic_int.h 中, 如下:

```
typedef struct
{
    uint32_t    m_Line;
    FunctionalState    m_Mask;
    FunctionalState    m_Rise;
    FunctionalState    m_Fall;
    uint32_t    m_Source;
} EINT_InitTypeDef;
```

表 14-2 INT 配置信息结构体说明

配置信息	描述
m_Line	外部中断编号, 取值为宏“外部中断编号”。

m_Mask	外部中断使能控制，取值为 TRUE 或 FALSE
m_Rise	外部中断上升沿中断使能，取值为 TRUE 或 FALSE。
m_Fall	外部中断下降沿中断使能，取值为 TRUE 或 FALSE。
m_Source	外部中断的中断源选择，取值为宏“外部中断源”。

14.3 INT 宏定义

INT 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，INT 其他相关宏定义详见文件 kf32a_basic_int.h 及函数参数描述。

14.4 INT 库函数

表 14-3 INT 固件库函数列表

序号	函数名	描述
1	INT_Get_Interrupt_Action	获取当前正在处理的中断向量编号。
2	INT_Get_Priority_Pending_Action	获取当前待响应的最高优先级中断向量编号。
3	INT_Priority_Base	设置优先级响应基级。
4	INT_Get_Priority_Base	获取优先级响应基级。
5	INT_Stack_Align_Config	设置中断自动堆栈对齐选择。
6	INT_Fault_Masking_Config	设置硬件错误中断屏蔽位。
7	INT_Get_Pre_Empty	判断是否一个悬起的中断将在下一步时进入活动状态。
8	INT_Get_Pending_Flag	判断当前是否有除 NMI 之外的中断挂起。
9	INT_Priority_Group_Config	设置优先级分组。
10	INT_Get_Priority_Group	获取优先级分组。
11	INT_All_Enable	全局可屏蔽中断使能位，该中断使能控制不包含复位/NMI/硬件错误中断。
12	INT_Interrupt_Enable	外设或内核中断使能控制，对于用户未定义的保留区中断向量，可能出现未知结果。
13	INT_Set_Systick_Flag	SYSTICK 中断标志软件置位。
14	INT_Set_PendSV_Flag	PendSV 中断标志软件置位。
15	INT_Get_Interrupt_Flag	获取外设或内核中断标志，对于用户未定义的保留区中断向量，可能出现未知结果。
16	INT_Clear_Interrupt_Flag	清外设或内核中断标志
17	INT_Interrupt_Priority_Config	外设或内核中断优先级配置，对于用户未定义的保留区中断向量，可能出现未知结果。
18	INT_Set_Interrupt_Priority	外设或内核中断优先级配置，对于用户未定义的保留区中断向量，可能出现未知结果。
19	INT_Stack_Delay_Enable	中断延时配置。

20	INT_Vector_Offset_Config	
21	INT_External_Configuration	外部中断(EINT)配置，并使能中断。
22	INT_External_Struct_Init	初始化外部中断(EINT)配置信息结构体。
23	INT_External_Mask_Enable	外部中断(EINT)使能配置。
24	INT_External_Rise_Enable	外部中断(EINT)上升沿中断使能配置。
25	INT_External_Fall_Enable	外部中断(EINT)下降沿中断使能配置。
26	INT_Get_External_Flag	获取外部中断(EINT)中断标志位。
27	INT_External_Clear_Flag	清除外部中断(EINT)中断标志位。
28	INT_External_Source_Enable	外部中断(EINT)中断源配置。

14.4.1 函数 INT_Get_Interrupt_Action

表 14-4 函数 INT_Get_Interrupt_Action

函数名	INT_Get_Interrupt_Action
函数原型	uint8_t INT_Get_Interrupt_Action (void)
功能描述	获取当前正在处理的中断向量编号。
输入参数 1	无
返回值	当前中断向量编号，7 位有效数据。
被调用函数	无

14.4.2 函数 INT_Get_Priority_Pending_Action

表 14-5 函数 INT_Get_Priority_Pending_Action

函数名	INT_Get_Priority_Pending_Action
函数原型	uint8_t INT_Get_Priority_Pending_Action (void)
功能描述	获取当前待响应的最高优先级中断向量编号。
输入参数 1	无
返回值	待响应的最高优先级中断向量编号，7 位有效数据。
被调用函数	无

14.4.3 函数 INT_Priority_Base

表 14-6 函数 INT_Priority_Base

函数名	INT_Priority_Base
函数原型	void INT_Priority_Base (uint8_t PriBase)
功能描述	设置优先级响应基级。
输入参数 1	PriBase: 优先级响应基级，取值 4 位数据。
返回值	无
被调用函数	Static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

14.4.4 函数 INT_Get_Priority_Base

表 14-7 函数 INT_Get_Priority_Base

函数名	INT_Get_Priority_Base
函数原型	uint8_t INT_Get_Priority_Base (void)
功能描述	获取优先级响应基级。
输入参数 1	无
返回值	优先级响应基级，4 位有效数据。
被调用函数	无

14.4.5 函数 INT_Stack_Align_Config

表 14-8 函数 INT_Stack_Align_Config

函数名	INT_Stack_Align_Config
函数原型	void INT_Stack_Align_Config (uint32_t StackAlign)
功能描述	设置中断自动堆栈对齐选择。
输入参数 1	StackAlign: 中断自动堆栈对齐选择，取值为： INT_STACK_DOUBLE_ALIGN: 中断自动堆栈使用双字对齐 INT_STACK_SINGLE_ALIGN: 中断自动堆栈使用单字对齐
返回值	无
被调用函数	无

14.4.6 函数 INT_Fault_Masking_Config

表 14-9 函数 INT_Fault_Masking_Config

函数名	INT_Fault_Masking_Config
函数原型	void INT_Fault_Masking_Config (FunctionalState NewState)
功能描述	设置硬件错误中断屏蔽位。
输入参数 1	NewState: 中断自动堆栈对齐选择，取值范围为：TRUE 或 FALSE。
返回值	无
被调用函数	无

14.4.7 函数 INT_Get_Pre_Empty

表 14-10 函数 INT_Get_Pre_Empty

函数名	INT_Get_Pre_Empty
函数原型	FlagStatus INT_Get_Pre_Empty (void)
功能描述	判断是否一个悬起的中断将在下一步时进入活动状态。

输入参数 1	无
返回值	1:进入活动状态；0:不进入活动状态。
被调用函数	无

14.4.8 函数 INT_Get_Pending_Flag

表 14-11 函数 INT_Get_Pending_Flag

函数名	INT_Get_Pending_Flag
函数原型	FlagStatus INT_Get_Pending_Flag (void)
功能描述	判断当前是否有除 NMI 之外的中断挂起。
输入参数 1	无
返回值	1:有除 NMI 之外的中断挂起；0:没有除 NMI 之外的中断挂起。
被调用函数	无

14.4.9 函数 INT_Priority_Group_Config

表 14-12 函数 INT_Priority_Group_Config

函数名	INT_Priority_Group_Config
函数原型	void INT_Priority_Group_Config (uint32_t PriorityGroup)
功能描述	设置优先级分组。
输入参数 1	PriorityGroup: 中断自动堆栈对齐选择，取值范围为： INT_PRIORITY_GROUP_3VS1: 抢占优先级为 3 位，子优先级为 1 位 INT_PRIORITY_GROUP_2VS2: 抢占优先级为 2 位，子优先级为 2 位 INT_PRIORITY_GROUP_1VS3: 抢占优先级为 1 位，子优先级为 3 位 INT_PRIORITY_GROUP_0VS4: 抢占优先级为 0 位，子优先级为 4 位
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

14.4.10 函数 INT_Get_Priority_Group

表 14-13 函数 INT_Get_Priority_Group

函数名	INT_Get_Priority_Group
函数原型	uint32_t INT_Get_Priority_Group (void)
功能描述	获取优先级分组。
输入参数 1	无
返回值	优先级分组位域配置，32 位有效数据。
被调用函数	无

14.4.11 函数 INT_All_Enable

表 14-14 函数 INT_All_Enable

函数名	INT_All_Enable
函数原型	void INT_All_Enable (FunctionalState NewState)
功能描述	全局可屏蔽中断使能位，该中断使能控制不包含复位/NMI/硬件错误中断。
输入参数 1	NewState: 全局可屏蔽中断使能，取值范围为：TRUE 或 FALSE。
返回值	无
被调用函数	无

14.4.12 函数 INT_Interrupt_Enable

表 14-15 函数 INT_Interrupt_Enable

函数名	INT_Interrupt_Enable
函数原型	void INT_Interrupt_Enable (InterruptIndex Peripheral, FunctionalState NewState)
功能描述	外设或内核中断使能控制，对于用户未定义的保留区中断向量，可能出现未知结果。
输入参数 1	输入 Peripheral: 外设或内核中断向量编号，取值范围为：枚举类型 InterruptIndex 中的外设中断向量编号。
输入参数 2	NewState: 外设或内核中断使能状态，取值范围为：TRUE 或 FALSE。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

14.4.13 函数 INT_Set_Systick_Flag

表 14-16 函数 INT_Set_Systick_Flag

函数名	INT_Set_Systick_Flag
函数原型	void INT_Set_Systick_Flag (void)
功能描述	YSTICK 中断标志软件置位。
输入参数 1	无
返回值	无
被调用函数	无

14.4.14 函数 INT_Set_PendSV_Flag

表 14-17 函数 INT_Set_PendSV_Flag

函数名	INT_Set_PendSV_Flag
函数原型	void INT_Set_PendSV_Flag (void)

功能描述	PendSV 中断标志软件置位。
输入参数 1	无
返回值	无
被调用函数	无

14.4.15 函数 INT_Get_Interrupt_Flag

表 14-18 函数 INT_Get_Interrupt_Flag

函数名	INT_Get_Interrupt_Flag
函数原型	FlagStatus INT_Get_Interrupt_Flag (InterruptIndex Peripheral)
功能描述	获取外设或内核中断标志，对于用户未定义的保留区中断向量，可能出现未知结果。
输入参数 1	Peripheral: 外设或内核中断向量编号，取值范围为：枚举类型 InterruptIndex 中的外设中断向量编号。
返回值	无
被调用函数	无

14.4.16 函数 INT_Clear_Interrupt_Flag

表 14-19 函数 INT_Clear_Interrupt_Flag

函数名	INT_Clear_Interrupt_Flag
函数原型	void INT_Clear_Interrupt_Flag (InterruptIndex Peripheral)
功能描述	清外设或内核中断标志
输入参数 1	Peripheral: 外设或内核中断向量编号，取值范围为：枚举类型 InterruptIndex 中的外设中断向量编号。
返回值	无
被调用函数	无

14.4.17 函数 INT_Interrupt_Priority_Config

表 14-20 函数 INT_Interrupt_Priority_Config

函数名	INT_Interrupt_Priority_Config
函数原型	void INT_Interrupt_Priority_Config (InterruptIndex Peripheral, uint32_t Preemption, uint32_t SubPriority)
功能描述	外设或内核中断优先级配置，对于用户未定义的保留区中断向量，可能出现未知结果。
输入参数 1	Peripheral: 外设或内核中断向量编号，取值范围为：枚举类型 InterruptIndex 中的外设中断向量编号。
输入参数 2	Preemption: 抢占优先级，同时满足 PRIGROUP 设置。
输入参数 3	SubPriority: 子优先级，同时满足 PRIGROUP 设置。

	(GROUP) == INT_PRIORITY_GROUP_3VS1 时: Preemption 取值范围为: 0~7, SubPriority 取值范围为: 0~1 (GROUP) == INT_PRIORITY_GROUP_2VS2 时: Preemption 取值范围为: 0~3, SubPriority 取值范围为: 0~3 (GROUP) == INT_PRIORITY_GROUP_1VS3 时: Preemption 取值范围为: 0~1, SubPriority 取值范围为: 0~7 (GROUP) == INT_PRIORITY_GROUP_0VS4 时: Preemption 取值范围为: 0~0, SubPriority 取值范围为: 0~15
返回值	无
被调用函数	无

14.4.18 函数 INT_Set_Interrupt_Priority

表 14-21 函数 INT_Set_Interrupt_Priority

函数名	INT_Set_Interrupt_Priority
函数原型	void INT_Set_Interrupt_Priority(InterruptIndex Peripheral, uint32_t Priority)
功能描述	外设或内核中断优先级配置, 对于用户未定义的保留区中断向量, 可能出现未知结果。
输入参数 1	Peripheral: 外设或内核中断向量编号, 取值范围为: 枚举类型 InterruptIndex 中的外设中断向量编号。
输入参数 2	Priority: 优先级, 同时满足 PRIGROUP 设置。Priority 取值范围为: 0~15
返回值	无
被调用函数	无

14.4.19 函数 INT_Stack_Delay_Enable

表 14-22 函数 INT_Stack_Delay_Enable

函数名	INT_Stack_Delay_Enable
函数原型	void INT_Stack_Delay_Enable (uint8_t IntDelay)
功能描述	中断延时配置。
输入参数 1	IntDelay: 中断延时控制, 取值 8 位数据。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

14.4.20 函数 INT_External_Configuration

表 14-23 函数 INT_External_Configuration

函数名	INT_External_Configuration
函数原型	void INT_External_Configuration (EINT_InitTypeDef* eintInitStruct)

功能描述	外部中断(EINT)配置，并使能中断。
输入参数 1	eintInitStruct: 外部中断配置信息结构体指针。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

14.4.21 函数 INT_External_Struct_Init

表 14-24 函数 INT_External_Struct_Init

函数名	INT_External_Struct_Init
函数原型	void INT_External_Struct_Init (EINT_InitTypeDef* eintInitStruct)
功能描述	初始化外部中断(EINT)配置信息结构体。
输入参数 1	eintInitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

14.4.22 函数 INT_External_Mask_Enable

表 14-25 函数 INT_External_Mask_Enable

函数名	INT_External_Mask_Enable
函数原型	void INT_External_Mask_Enable (uint32_t EintMask, FunctionalState NewState)
功能描述	外部中断(EINT)使能配置。
输入参数 1	EintMask: 外部中断编号掩码，取值为宏 INT_EINTMASK_EINTM0 至 INT_EINTMASK_EINTM31 的位或组合。
输入参数 2	NewState: 外部中断使能请求，取值范围为：TRUE 或 FALSE。
返回值	无
被调用函数	无

14.4.23 函数 INT_External_Rise_Enable

表 14-26 函数 INT_External_Rise_Enable

函数名	INT_External_Rise_Enable
函数原型	void INT_External_Rise_Enable (uint32_t EintMask, FunctionalState NewState)
功能描述	外部中断(EINT)上升沿中断使能配置。
输入参数 1	EintMask: 外部中断编号掩码，取值为宏 INT_EINTMASK_EINTM0 至 INT_EINTMASK_EINTM31 的位或组合。
输入参数 2	NewState: 外部中断上升沿中断使能，取值范围为：TRUE 或 FALSE。
返回值	无

被调用函数	无
-------	---

14.4.24 函数 INT_External_Fall_Enable

表 14-27 函数 INT_External_Fall_Enable

函数名	INT_External_Fall_Enable
函数原型	void INT_External_Fall_Enable (uint32_t EintMask, FunctionalState NewState)
功能描述	外部中断(EINT)下降沿中断使能配置。
输入参数 1	EintMask: 外部中断编号掩码, 取值为宏 INT_EINTMASK_EINTM0 至 INT_EINTMASK_EINTM31 的位或组合。
输入参数 2	NewState: 外部中断下降沿中断使能, 取值范围为: TRUE 或 FALSE。
返回值	无
被调用函数	无

14.4.25 函数 INT_Get_External_Flag

表 14-28 函数 INT_Get_External_Flag

函数名	INT_Get_External_Flag
函数原型	FlagStatus INT_Get_External_Flag (uint32_t EintNum)
功能描述	获取外部中断(EINT)中断标志位。
输入参数 1	EintNum: 外部中断编号, 取值为宏 INT_EXTERNAL_INTERRUPT_0 至 INT_EXTERNAL_INTERRUPT_31 中的一个, 即 0~31。
返回值	外部中断(EINT)中断标志, 0: 没有发生外部中断, 1: 发生外部中断。
被调用函数	无

14.4.26 函数 INT_External_Clear_Flag

表 14-29 函数 INT_External_Clear_Flag

函数名	INT_External_Clear_Flag
函数原型	void INT_External_Clear_Flag (uint32_t EintNum)
功能描述	清除外部中断(EINT)中断标志位。
输入参数 1	EintNum: 外部中断编号, 取值为宏 INT_EXTERNAL_INTERRUPT_0 至 INT_EXTERNAL_INTERRUPT_31 中的一个, 即 0~31。
返回值	无
被调用函数	无

14.4.27 函数 INT_External_Source_Enable

表 14-30 函数 INT_External_Source_Enable

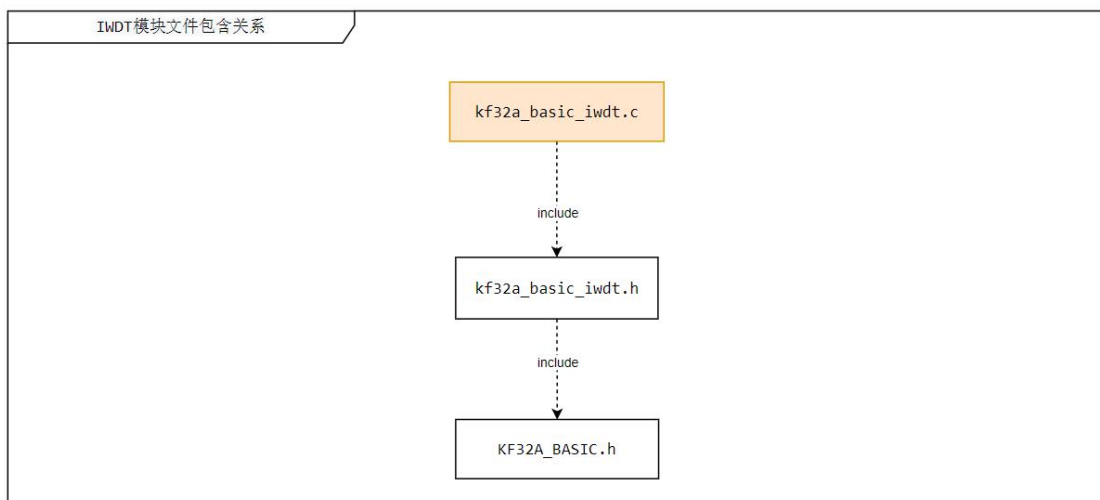
函数名	INT_External_Source_Enable
函数原型	void INT_External_Source_Enable (uint32_t EintNum, uint32_t PeripheralSource)
功能描述	外部中断(EINT)中断源配置。
输入参数 1	EintNum: 外部中断编号, 取值为宏 INT_EXTERNAL_INTERRUPT_0 至 INT_EXTERNAL_INTERRUPT_31 中的一个, 即 0~15。
输入参数 2	PeripheralSource: 外设中断线的中断输入源, 取值范围为: INT_EXTERNAL_SOURCE_PA INT_EXTERNAL_SOURCE_PB INT_EXTERNAL_SOURCE_PC INT_EXTERNAL_SOURCE_PD INT_EXTERNAL_SOURCE_PE INT_EXTERNAL_SOURCE_PF INT_EXTERNAL_SOURCE_PG INT_EXTERNAL_SOURCE_PH INT_EXTERNAL_SOURCE_PVD INT_EXTERNAL_SOURCE_RTC INT_EXTERNAL_SOURCE_TAMPER INT_EXTERNAL_SOURCE_ALARMCLK INT_EXTERNAL_SOURCE_AES INT_EXTERNAL_SOURCE_EINT21TO31
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

15 独立看门狗（IWDT）

KF32A 系列芯片带有硬件独立看门狗模块(IWDT)，用于检测 and 解决软件错误引起的故障。当故障发生时，看门狗计数器达到设定的超时值会产生一次系统复位。在设定的超时值内，无故障发生，需要进行喂狗操作，使看门狗计数器清零。独立看门狗使用的工作时钟为 32.678KHz 的内部低频时钟。

15.1 文件引用关系

图 15-1 IWDT 模块文件包含关系



15.2 IWDT 寄存器结构体

IWDT 寄存器结构，IWDT_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct IWDT_MemMap
{
    volatile uint32_t    CTL;
    volatile uint32_t    FD;
}IWDT_SFRmap;
  
```

表 15-1 IWDT 寄存器结构说明

寄存器	描述
CTL	独立看门狗控制寄存器，偏移:0x00
FD	独立看门狗喂狗寄存器，偏移:0x04

15.3 IWDG 宏定义

ADC 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，ADC 其他相关宏定义详见文件 kf32a_basic_iwdt.h 及函数参数描述。

15.4 IWDG 库函数

表 15-2 IWDG 固件库函数列表

函数名	描述
IWDG_Prescaler_Config	独立看门狗预分频配置
IWDG_Overflow_Config	独立看门狗溢出值配置
IWDG_Enable	独立看门狗使能与禁能
IWDG_Feed_The_Dog	独立看门狗喂狗操作

15.4.1 函数 IWDG_Prescaler_Config

表 15-3 函数 IWDG_Prescaler_Config

函数名	IWDG_Prescaler_Config
函数原型	void IWDG_Prescaler_Config (uint32_t Prescaler)
功能描述	设置独立看门狗预分频值
输入参数 1	Clock:系统节拍定时器时钟源选择，取值范围： IWDG_PRESCALER_32: 32 分频 IWDG_PRESCALER_64: 64 分频 IWDG_PRESCALER_128: 128 分频 IWDG_PRESCALER_256: 256 分频 IWDG_PRESCALER_512: 512 分频 IWDG_PRESCALER_1024: 1024 分频 IWDG_PRESCALER_2048: 2048 分频 IWDG_PRESCALER_4096: 4096 分频 IWDG_PRESCALER_8192: 8192 分频 IWDG_PRESCALER_16384: 16384 分频 IWDG_PRESCALER_32768: 32768 分频 IWDG_PRESCALER_65536: 65536 分频
返回值	无
被调用函数	无

15.4.2 函数 IWDG_Overflow_Config

表 15-4 函数 IWDG_Overflow_Config

函数名	IWDG_Overflow_Config
-----	----------------------

函数原型	void IWDT_Overflow_Config (uint32_t Overflow)
功能描述	设置独立看门狗溢出值，溢出时复位
输入参数 1	Overflow: 独立看门狗溢出值，取值范围：0~0xFFF
返回值	无
被调用函数	无

15.4.3 函数 IWDT_Enable

表 15-5 函数 IWDT_Overflow_Config

函数名	IWDT_Enable
函数原型	void IWDT_Enable (FunctionalState NewState)
功能描述	设置独立看门狗使能与禁能
输入参数 1	NewState: 独立看门狗使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

15.4.4 函数 IWDT_Feed_The_Dog

表 15-6 函数 WDT_Feed_The_Dog

函数名	IWDT_Feed_The_Dog
函数原型	void IWDT_Feed_The_Dog (void)
功能描述	独立看门狗喂狗，清看门狗计数器
输入参数 1	无
返回值	无
被调用函数	无

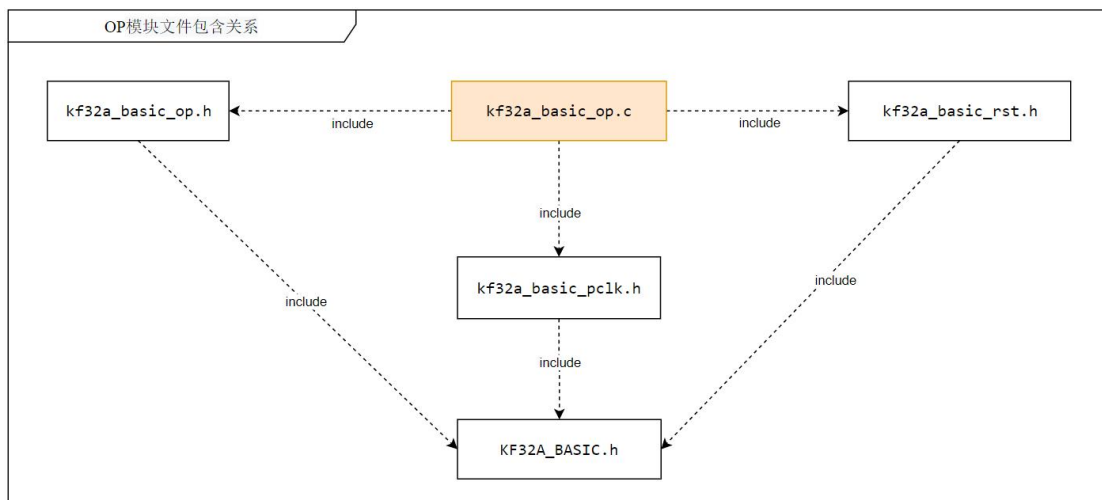
16 可编程增益放大器模块（OP）

KungFu32 内核单片机内置 3 个可编程增益运算放大器模块（OP）。其主要特点如下：

正负端多输入端口可选；OP 增益可调（10X/20X/40X/80X）；OP 放大模式下的高带宽；输出可以直连 AD，进行采样；备注：本系统暂不支持 OP0/OP1 的单位增益缓冲器应用方式

16.1 文件引用关系

图 16-1 OP 模块文件包含关系



16.2 OP 寄存器结构

OP 寄存器结构，OP_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct OP_MemMap
{
    volatile uint32_t    CAL;
    volatile uint32_t    CTL0;
    volatile uint32_t    CTL1;
}OP_SFRmap;
    
```

表 16-1 OP 寄存器结构说明

寄存器	描述
CAL	运算放大器校准寄存器，偏移:0x00
CAL0	运算放大器校准寄存器 0，偏移:0x04
CAL1	运算放大器校准寄存器 1，偏移:0x08

16.3 OP 宏定义

OP 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，OP 其他相关宏定义详见文件 kf32a_basic_op.h 及函数参数描述。

16.4 OP 库函数

表 16-4 OP 固件库函数列表

序号	函数名	描述
1	OP_Reset	OP 外设复位，使能外设时钟
2	OP_CAL_Configure	OP 校准值
3	OP_GAIN_SELSECT	OP 增益选择
4	OP3_POSITIVE_INPUT_SELSECT	运放 3 正端输入选择
5	OP_OUTPUT_EN	运放输出使能选择
6	OP_MODULE_EN	运放模块使能选择

16.4.1 函数 OP_Reset

表 16-2 函数 OP_Reset

函数名	OP_Reset
函数原型	void OP_Reset(void)
功能描述	OP 外设复位，使能外设时钟
返回值	无
被调用函数	RST_CTL3_Peripheral_Reset_Enable(RST_CTL3_OPRST, TRUE); RST_CTL3_Peripheral_Reset_Enable(RST_CTL3_OPRST, FALSE); PCLK_CTL3_Peripheral_Clock_Enable(PCLK_CTL3_OPCLKEN, TRUE);

16.4.2 函数 OP_CAL_Configure

表 16-3 函数 OP_CAL_Configure

函数名	OP_CAL_Configure
函数原型	void OP_CAL_Configure(uint32_t OPx , uint32_t CAL_VALUE)
功能描述	OP 校准值
输入参数 1	OP0/OP1/OP2/OP3
输入参数 2	CAL_VALUE 范围位宽 6bit 位，值 0x00-0x3f
返回值	无
被调用函数	无

16.4.3 函数 OP_GAIN_SELECT

表 16-4 函数 OP_GAIN_SELECT

函数名	OP_GAIN_SELECT
函数原型	void OP_GAIN_SELECT(uint32_t OPx, uint32_t GAIN_VALUE)
功能描述	OP 增益选择
输入参数 1	OP0/OP1/OP2/OP3
输入参数 2	CAL_VALUE 范围:GAIN_10/ GAIN_20/ GAIN_40/ GAIN_80
返回值	无
被调用函数	无

16.4.4 函数 OP3_POSITIVE_INPUT_SELECT

表 16-5 函数 OP3_POSITIVE_INPUT_SELECT

函数名	OP3_POSITIVE_INPUT_SELECT
函数原型	void OP3_POSITIVE_INPUT_SELECT(uint32_t SEL)
功能描述	运放 3 正端输入选择
输入参数 1	OP3_POSITIVE_INPUT_AVDD OP3_POSITIVE_INPUT_VREF2V
返回值	无
被调用函数	无

16.4.5 函数 OP_OUTPUT_EN

表 16-6 函数 OP_OUTPUT_EN

函数名	OP_OUTPUT_EN
函数原型	void OP_OUTPUT_EN(uint32_t OPx, FunctionalState NewState)
功能描述	运放输出使能选择
输入参数 1	OP0/OP1/OP2/OP3
输入参数 2	TRUE/FALSE
返回值	无
被调用函数	无

16.4.6 函数 OP_MODULE_EN

表 16-7 函数 OP_MODULE_EN

函数名	OP_MODULE_EN
函数原型	void OP_MODULE_EN(uint32_t OPx, FunctionalState NewState)
功能描述	运放模块使能选择
输入参数 1	OP0/OP1/OP2/OP3
输入参数 2	TRUE/FALSE

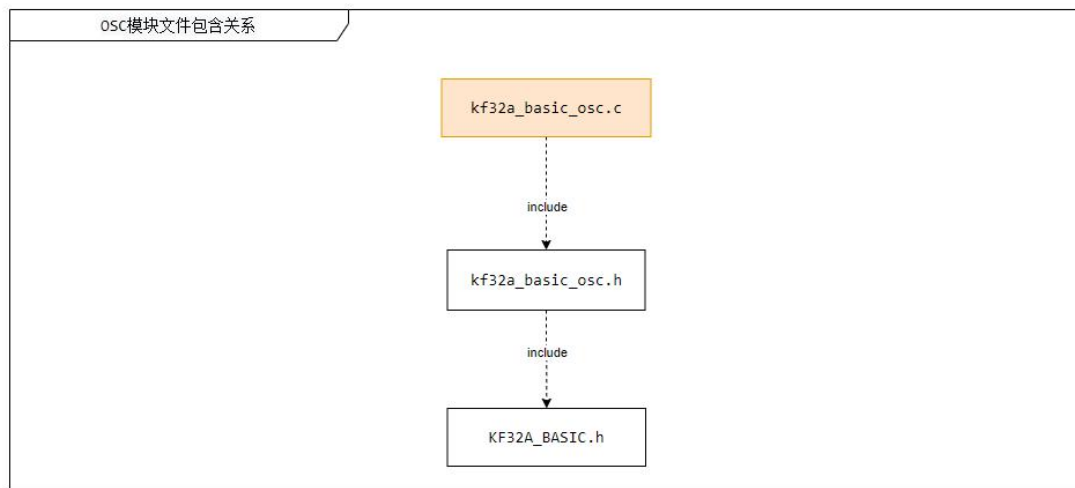
返回值	无
被调用函数	无

17 振荡器（OSC）

KF32A 系列 MCU 提供 6 种基础时钟振荡器选择，分别为内部高频（INTHF）、内部低频（INTLF）、外部高频（EXTHF）、外部低频（EXTLF）、内部的 PLL 和低功耗 4M 时钟 LP4M。内部的 PLL 可以将内部高频（INTHF）和外部高频（EXTHF）的输出时钟倍频，提供更高频率的工作时钟选择，作为系统和外设工作需要的基础时钟。通过寄存器配置，可以从 6 种振荡器中得到 3 种系统和外设运行时需要的时钟源：系统主时钟（SCLK）、低频外设时钟（LFCLK）和高速外设时钟（HFCLK）满足不同的需要。

17.1 文件引用关系

图 17-1 OSC 模块文件包含关系



17.2 OSC 寄存器结构

OSC 寄存器结构，OSC_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct OSC_MemMap {
    volatile uint32_t    CTL0;
    volatile uint32_t    CTL1;
    volatile uint32_t    INT;
    volatile uint32_t    CTL2;
    volatile uint32_t    HFOSCCAL0;
    volatile uint32_t    HFOSCCAL1;
}OSC_SFRmap;
  
```

表 17-1 OSC 寄存器结构说明

寄存器	描述
CTL0	振荡器控制寄存器 0，偏移:0x00
CTL1	振荡器控制寄存器 1，偏移:0x04

INT	振荡器中断控制寄存器, 偏移:0x08
CTL2	振荡器控制寄存器 2, 偏移:0x0C
HFOSCCAL0	高频振荡器校准寄存器 0, 偏移:0x10
HFOSCCAL1	高频振荡器校准寄存器 1, 偏移:0x14

PLL 寄存器结构, PLL_SFRmap, 定义于文件 KF32A_BASIC.h 中, 如下:

```
typedef struct PLL_MemMap {
    volatile uint32_t    CTL;
}PLL_SFRmap;
```

表 17-2 PLL 寄存器结构说明

寄存器	描述
CTL	PLL 控制寄存器 0, 偏移:0x00

17.3 OSC 宏定义

OSC 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, OSC 其他相关宏定义详见文件 kf32a_basic_osc.h 及函数参数描述。

17.4 OSC 库函数

表 17-3 OSC 固件库函数列表

序号	函数名	描述
1	OSC_SCLK_Configuration	配置振荡器系统主时钟 SCLK
2	OSC_HFCK_Configuration	配置振荡器高频外设时钟 HFCLK
3	OSC_LFCK_Configuration	配置振荡器低频外设时钟 LFCLK
4	OSC_CK48M_Configuration	配置振荡器 48MHz 时钟 CK48M
5	OSC_Struct_Init	初始化振荡器配置信息结构体
6	OSC_LFCK_Division_Config	配置低频外设时钟分频选择
7	OSC_HFCK_Division_Config	配置高频外设时钟分频选择
8	OSC_SCK_Division_Config	配置主时钟 SCLK 分频选择
9	OSC_PLL_Input_Source_Config	配置 PLL 输入时钟源选择
10	OSC_HFCK_Source_Config	配置高频外设时钟源选择
11	OSC_HFCK_Enable	配置高频外设时钟允许使能
12	OSC_LFCK_Source_Config	配置低频外设时钟源选择
13	OSC_LFCK_Enable	配置低频外设时钟允许使能
14	OSC_SCK_Source_Config	配置 SCLK 主时钟选择
15	OSC_Backup_Write_Read_Enable	配置备份区寄存器读写允许
16	OSC_SCLK_Output_Enable	配置系统时钟输出使能
17	OSC_SCLK_Output_Select	配置从引脚 CLKOUT 输出时钟选择
18	OSC_SCLK_Output_Division_Config	配置 SCLK 输出时钟分频选择

19	OSC_Clock_Failure_Check_Enable	配置时钟故障检测使能
20	OSC_CK48M_Division_Config	配置 CK48M 时钟分频选择
21	OSC_CK48M_Source_Config	配置 48M 时钟输入时钟源选择
22	OSC_CK48M_Enable	配置 48M 时钟输入时钟允许使能
23	OSC_PLL_Multiple_Value_Select	配置 PLL 倍频选择
24	OSC_PLL_RST	PLL 复位控制
25	OSC_PLL_Start_Delay_Config	配置 PLL 启动延迟控制
26	OSC_EXTHF_Start_Delay_Config	配置外部高频启动延迟控制
27	OSC_EXTLF_Start_Delay_Config	配置外部低频启动延迟控制
28	OSC_PLL_Software_Enable	配置 PLL 软件使能
29	OSC_EXTHF_Software_Enable	配置外部高频振荡器软件使能
30	OSC_EXTLF_Software_Enable	配置外部低频振荡器软件使能
31	OSC_INTHF_Software_Enable	配置内部高频振荡器软件使能
32	OSC_INTLF_Software_Enable	配置内部低频振荡器软件使能
33	OSC_Zero_Drift_Config	配置零温漂的绝对值校准权重位
34	OSC_Positive_Drift_Config	配置内部高频振荡器的正温调节
35	OSC_Negative_Drift_Config	配置内部高频振荡器的负温调节
36	OSC_Current_Gain_Config	配置电流增益选择
37	OSC_High_Speed_Enable	配置高速选择
38	OSC_External_Input_Enable	配置外部高频晶振输入使能
39	OSC_Feedback_Resistance_Config	配置反馈电阻调节
40	OSC_PLL_Zero_Source_Enable	配置 PLL 零温漂电流源输出到 IO 口使能
41	OSC_PLL_Vref2_Enable	配置 PLL 内部参考电压 2V 输出到 IO 口的使能
42	OSC_PLL_Vref0p5_Enable	配置 PLL 内部低功耗 BG 参考 0.5V 到内部 buffer 的使能
43	OSC_PLL_Vref1p2_Enable	配置 PLL 主 BG 到内部 buffer 的通道使能
44	OSC_PLL_Low_Power_20nA_Enable	配置 PLL 内部低功耗 BG 的电流源 20nA 到 IO 口的选通使能
45	OSC_PLL_Vref1p14_Enable	配置 PLL 内部低功耗参考电压 1.14V 到内部 buffer 的选通使能
46	OSC_PLL_Low_Power_100nA_Enable	配置 PLL 内部低功耗偏置电路的电流源 100nA 到 IO 口的选通使能
47	OSC_PLL_LDO_Enable	配置 PLL 内部高频 LDO 到内部 buffer 的选通使能
48	OSC_PLL_INT_Enable	配置 PLL 中断使能
49	OSC_EXTHF_INT_Enable	配置外部高频中断使能
50	OSC_EXTLF_INT_Enable	配置外部低频中断使能
51	OSC_INTHF_INT_Enable	配置内部高频中断使能
52	OSC_INTLF_INT_Enable	配置内部低频中断使能
53	OSC_Get_Clock_Failure_INT_Flag	读取时钟故障标志

54	OSC_Get_PLL_INT_Flag	读取 PLL 中断标志
55	OSC_Get_EXTHF_INT_Flag	读取外部高频中断标志
56	OSC_Get_EXTLF_INT_Flag	读取外部低频中断标志
57	OSC_Get_INTHF_INT_Flag	读取内部高频中断标志
58	OSC_Get_INTLF_INT_Flag	读取内部低频中断标志
59	OSC_Get_LP4MIF_INT_Flag	读取 LP4M 中断标志

17.4.1 函数 OSC_SCLK_Configuration

表 17-4 函数 OSC_SCLK_Configuration

函数名	OSC_SCLK_Configuration
函数原型	void OSC_SCLK_Configuration (OSC_InitTypeDef* oscInitStruct)
功能描述	配置振荡器系统主时钟 SCLK
输入参数 1	oscInitStruct: 振荡器(OSC)配置信息结构体指针
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.2 函数 OSC_HFCK_Configuration

表 17-5 函数 OSC_HFCK_Configuration

函数名	OSC_HFCK_Configuration
函数原型	void OSC_HFCK_Configuration (OSC_InitTypeDef* oscInitStruct)
功能描述	配置振荡器高频外设时钟 HFCLK
输入参数 1	oscInitStruct: 振荡器(OSC)配置信息结构体指针
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.3 函数 OSC_LFCK_Configuration

表 17-6 函数 OSC_LFCK_Configuration

函数名	OSC_LFCK_Configuration
函数原型	void OSC_LFCK_Configuration (OSC_InitTypeDef* oscInitStruct)
功能描述	配置振荡器低频外设时钟 LFCLK
输入参数 1	oscInitStruct: 振荡器(OSC)配置信息结构体指针
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.4 函数 OSC_CK48M_Configuration

表 17-7 函数 OSC_CK48M_Configuration

函数名	OSC_CK48M_Configuration
函数原型	void OSC_CK48M_Configuration (OSC_InitTypeDef* oscInitStruct)
功能描述	配置振荡器 48MHz 时钟 CK48M
输入参数 1	oscInitStruct: 振荡器(OSC)配置信息结构体指针
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.5 函数 OSC_Struct_Init

表 17-8 函数 OSC_Struct_Init

函数名	OSC_Struct_Init
函数原型	void OSC_Struct_Init (OSC_InitTypeDef* oscInitStruct)
功能描述	初始化振荡器配置信息结构体
输入参数 1	oscInitStruct: 指向待初始化的结构体指针
返回值	无
被调用函数	无

17.4.6 函数 OSC_LFCK_Division_Config

表 17-9 函数 OSC_LFCK_Division_Config

函数名	OSC_LFCK_Division_Config
函数原型	void OSC_LFCK_Division_Config (uint32_t LFDivision)
功能描述	配置低频外设时钟分频选择
输入参数 1	LFDivision: 低频外设时钟分频选择, 取值范围为: LFCK_DIVISION_1: 1/1 分频 LFCK_DIVISION_2: 1/2 分频 LFCK_DIVISION_4: 1/4 分频 LFCK_DIVISION_8: 1/8 分频 LFCK_DIVISION_16: 1/16 分频 LFCK_DIVISION_32: 1/32 分频 LFCK_DIVISION_64: 1/64 分频 LFCK_DIVISION_128: 1/128 分频
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.7 函数 OSC_HFCK_Division_Config

表 17-10 函数 OSC_HFCK_Division_Config

函数名	OSC_HFCK_Division_Config
函数原型	void OSC_HFCK_Division_Config (uint32_t HFDivision)
功能描述	配置高频外设时钟分频选择
输入参数 1	HFDivision: 高频外设时钟分频选择, 取值范围为: HFCK_DIVISION_1: 1/1 分频 HFCK_DIVISION_2: 1/2 分频 HFCK_DIVISION_4: 1/4 分频 HFCK_DIVISION_8: 1/8 分频 HFCK_DIVISION_16: 1/16 分频 HFCK_DIVISION_32: 1/32 分频 HFCK_DIVISION_64: 1/64 分频 HFCK_DIVISION_128: 1/128 分频 HFCK_DIVISION_256: 1/256 分频 HFCK_DIVISION_512: 1/512 分频
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.8 函数 OSC_SCK_Division_Config

表 17-11 函数 OSC_SCK_Division_Config

函数名	OSC_SCK_Division_Config
函数原型	void OSC_SCK_Division_Config (uint32_t SclkDivision)
功能描述	配置主时钟 SCLK 分频选择
输入参数 1	SclkDivision: 主时钟 SCLK 分频选择, 取值范围为: SCLK_DIVISION_1: 1/1 分频 SCLK_DIVISION_2: 1/2 分频 SCLK_DIVISION_4: 1/4 分频 SCLK_DIVISION_8: 1/8 分频 SCLK_DIVISION_16: 1/16 分频 SCLK_DIVISION_32: 1/32 分频 SCLK_DIVISION_64: 1/64 分频 SCLK_DIVISION_128: 1/128 分频
返回值	无
被调用函数	无

17.4.9 函数 OSC_PLL_Input_Source_Config

表 17-12 函数 OSC_PLL_Input_Source_Config

函数名	OSC_PLL_Input_Source_Config
函数原型	void OSC_PLL_Input_Source_Config (uint32_t NewState)
功能描述	配置 PLL 输入时钟源选择
输入参数 1	NewState: PLL 输入时钟源选择, 取值范围为: PLL_INPUT_INTHF: 选择 INTHF 作为 PLL 输入时钟 PLL_INPUT_EXTHF: 选择 EXTHF 作为 PLL 输入时钟
返回值	无
被调用函数	无

17.4.10 函数 OSC_HFCK_Source_Config

表 17-13 函数 OSC_HFCK_Source_Config

函数名	OSC_HFCK_Source_Config
函数原型	void OSC_HFCK_Source_Config (uint32_t HFSource)
功能描述	配置高频外设时钟源选择
输入参数 1	HFSource: 高频外设时钟源选择, 取值范围为: HFCK_SOURCE_INTHF: 选择 INTHF 作为 HFCLK 时钟 HFCK_SOURCE_EXTHF: 选择 EXTHF 作为 HFCLK 时钟 HFCK_SOURCE_PLL: 选择 PLL 输出作为 HFCLK 时钟 HFCK_SOURCE_PLL2: 选择 PLL2 输出作为 HFCLK 时钟 HFCK_SOURCE_PLL1: 选择 PLL1 输出作为 HFCLK 时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.11 函数 OSC_HFCK_Enable

表 17-14 函数 OSC_HFCK_Enable

函数名	OSC_HFCK_Enable
函数原型	void OSC_HFCK_Enable (FunctionalState NewState)
功能描述	配置高频外设时钟允许使能
输入参数 1	NewState: 高频外设时钟允许使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.12 函数 OSC_LFCK_Source_Config

表 17-15 函数 OSC_HFCK_Enable

函数名	OSC_HFCK_Enable
函数原型	void OSC_LFCK_Source_Config (uint32_t NewState)
功能描述	配置低频外设时钟源选择
输入参数 1	NewState: 低频外设时钟源选择, 取值范围为: LFCK_INPUT_INTLF: 选择 INTLF 作为 LFCLK 时钟, LFCK_INPUT_EXTLF: 选择 EXTLF 作为 LFCLK 时钟
返回值	无
被调用函数	无

17.4.13 函数 OSC_LFCK_Enable

表 17-16 函数 OSC_LFCK_Enable

函数名	OSC_LFCK_Enable
函数原型	void OSC_LFCK_Enable (FunctionalState NewState)
功能描述	配置低频外设时钟允许使能
输入参数 1	NewState: 低频外设时钟允许使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.14 函数 OSC_SCK_Source_Config

表 17-17 函数 OSC_SCK_Source_Config

函数名	OSC_SCK_Source_Config
函数原型	void OSC_SCK_Source_Config (uint32_t SclkSource)
功能描述	配置 SCLK 主时钟选择
输入参数 1	SclkSource: SCLK 主时钟选择, 取值范围为: SCLK_SOURCE_INTHF: 选择 INTHF 作为系统时钟 SCLK_SOURCE_INTLF: 选择 INTLF 作为系统时钟 SCLK_SOURCE_EXTHF: 选择 EXTHF 作为系统时钟 SCLK_SOURCE_EXTLF: 选择 EXTLF 作为系统时钟 SCLK_SOURCE_PLL: 选择 PLL 输出作为系统时钟 SCLK_SOURCE_LP4M: 选择 LP4M 输出作为系统时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.15 函数 OSC_Backup_Write_Read_Enable

表 17-18 函数 OSC_Backup_Write_Read_Enable

函数名	OSC_Backup_Write_Read_Enable
函数原型	void OSC_Backup_Write_Read_Enable (FunctionalState NewState)
功能描述	配置备份区寄存器读写允许
输入参数 1	NewState: 备份区寄存器读写允许状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.16 函数 OSC_SCLK_Output_Enable

表 17-19 函数 OSC_SCLK_Output_Enable

函数名	OSC_SCLK_Output_Enable
函数原型	void OSC_SCLK_Output_Enable (FunctionalState NewState)
功能描述	配置系统时钟输出使能
输入参数 1	NewState: 系统时钟输出使能状态, 取值范围为: TRUE 或 FALSE。
返回值	无
被调用函数	无

17.4.17 函数 OSC_SCLK_Output_Select

表 17-20 函数 OSC_SCLK_Output_Select

函数名	OSC_SCLK_Output_Select
函数原型	void OSC_SCLK_Output_Select (uint32_t SclkSource)
功能描述	配置从引脚 CLKOUT 输出时钟选择
输入参数 1	SclkSource: 输出时钟选择, CLKOUT_SCLK: 选择 SCLK 作为输出时钟 CLKOUT_EXTLTF: 选择 EXTLF 作为输出时钟 CLKOUT_EXTHF: 选择 EXTHF 作为输出时钟 CLKOUT_INTLF: 选择 INTLF 作为输出时钟 CLKOUT_INTHF: 选择 INTHF 作为输出时钟 CLKOUT_PLL: 选择 PLL 作为输出时钟 CLKOUT_LP4M: 选择 LP4M 作为输出时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.18 函数 OSC_SCLK_Output_Division_Config

表 17-21 函数 OSC_SCLK_Output_Division_Config

函数名	OSC_SCLK_Output_Division_Config
函数原型	void OSC_SCLK_Output_Division_Config (uint32_t OutputDivision)
功能描述	配置 SCLK 输出时钟分频选择
输入参数 1	OutputDivision: SCLK 输出时钟分频选择位，取值范围为： SCLK_DIVISION_1: 1/1 分频 SCLK_DIVISION_2: 1/2 分频 SCLK_DIVISION_4: 1/4 分频 SCLK_DIVISION_8: 1/8 分频 SCLK_DIVISION_16: 1/16 分频 SCLK_DIVISION_32: 1/32 分频 SCLK_DIVISION_64: 1/64 分频 SCLK_DIVISION_128: 1/128 分频
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.19 函数 OSC_Clock_Failure_Check_Enable

表 17-22 函数 OSC_Clock_Failure_Check_Enable

函数名	OSC_Clock_Failure_Check_Enable
函数原型	void OSC_Clock_Failure_Check_Enable (FunctionalState NewState)
功能描述	配置时钟故障检测使能
输入参数 1	NewState: 时钟故障检测使能状态，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.20 函数 OSC_CK48M_Division_Config

表 17-23 函数 OSC_CK48M_Division_Config

函数名	OSC_CK48M_Division_Config
函数原型	void OSC_CK48M_Division_Config (uint32_t CK48MDivision)
功能描述	配置 CK48M 时钟分频选择
输入参数 1	CK48MDivision: CK48M 时钟分频选择，取值范围为： CK48M_DIVISION_1: 1/1 分频 CK48M_DIVISION_2: 1/2 分频 CK48M_DIVISION_4: 1/4 分频
返回值	无
被调用函数	无

17. 4. 21 函数 OSC_CK48M_Source_Config

表 17-24 函数 OSC_CK48M_Source_Config

函数名	OSC_CK48M_Source_Config
函数原型	void OSC_CK48M_Source_Config (uint32_t CK48MSource)
功能描述	配置 48M 时钟输入时钟源选择
输入参数 1	CK48MSource: 48M 时钟输入时钟源选择, 取值范围为: CK48M_SOURCE_INTHF: 选择 INTHF 作为 CK48M 输入时钟 CK48M_SOURCE_EXTHF: 选择 EXTHF 作为 CK48M 输入时钟 CK48M_SOURCE_PLL: 选择 PLL 输出作为 CK48M 输入时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17. 4. 22 函数 OSC_CK48M_Enable

表 17-25 函数 OSC_CK48M_Enable

函数名	OSC_CK48M_Enable
函数原型	void OSC_CK48M_Enable (FunctionalState NewState)
功能描述	配置 48M 时钟输入时钟允许使能
输入参数 1	NewState: 48M 时钟输入时钟允许使能状态, 取值范围为: TRUE 或 FALSE。
返回值	无
被调用函数	无

17. 4. 23 函数 OSC_PLL_Multiple_Value_Select

表 17-26 函数 OSC_PLL_Multiple_Value_Select

函数名	OSC_PLL_Multiple_Value_Select
函数原型	void OSC_PLL_Multiple_Value_Select (uint32_t PLLmultiple_M,uint32_t PLLmultiple_N,uint32_t PLLmultiple_NO)
功能描述	配置 PLL 倍频选择
输入参数 1	PLLmultiple: PLL 倍频, 取值范围为: PLLmultiple_M 0X04-0X3FFF 范围(M 要大于等于 4, 并且满足 $200\text{MHZ} < (M * \text{输入参考频率} / N) < 400\text{MHZ}$)
输入参数 2	PLLmultiple: PLL 倍频, 取值范围为: PLLmultiple_N 0X01-0XF 范围, 并且满足 $1\text{MHZ} < (\text{输入参考频率} / N) < 50\text{MHZ}$
输入参数 2	PLLmultiple: PLL 倍频, 取值范围为:

	PLLmultiple_NO 取值, 1/2/4/8 选其中 1 个
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.24 函数 OSC_PLL_RST

表 17-27 函数 OSC_PLL_RST

函数名	OSC_PLL_RST
函数原型	void OSC_PLL_RST (void)
功能描述	PLL 复位控制
输入参数 1	无
返回值	无
被调用函数	无

17.4.25 函数 OSC_PLL_Start_Delay_Config

表 17-28 函数 OSC_PLL_Start_Delay_Config

函数名	OSC_PLL_Start_Delay_Config
函数原型	void OSC_PLL_Start_Delay_Config (uint32_t PLLDelay)
功能描述	配置 PLL 启动延迟控制
输入参数 1	PLLDelay: PLL 启动延迟控制, 取值范围为: PLL_START_DELAY_64: 延迟 64 个时钟 PLL_START_DELAY_128: 延迟 128 个时钟 PLL_START_DELAY_256: 延迟 256 个时钟 PLL_START_DELAY_512: 延迟 512 个时钟 PLL_START_DELAY_1024: 延迟 1024 个时钟 PLL_START_DELAY_2048: 延迟 2048 个时钟 PLL_START_DELAY_4096: 延迟 4096 个时钟 PLL_START_DELAY_8192: 延迟 8192 个时钟 PLL_START_DELAY_16384: 延迟 16384 个时钟 PLL_START_DELAY_32768: 延迟 32768 个时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.26 函数 OSC_EXTHF_Start_Delay_Config

表 17-29 函数 OSC_EXTHF_Start_Delay_Config

函数名	OSC_EXTHF_Start_Delay_Config
函数原型	void OSC_EXTHF_Start_Delay_Config (uint32_t ExternalDelay)

功能描述	配置外部高频启动延迟控制
输入参数 1	ExternalDelay: 外部高频启动延迟控制, 取值范围为: EXT_START_DELAY_0: 延迟 0 个时钟 EXT_START_DELAY_2: 延迟 2 个时钟 EXT_START_DELAY_4: 延迟 4 个时钟 EXT_START_DELAY_8: 延迟 8 个时钟 EXT_START_DELAY_16: 延迟 16 个时钟 EXT_START_DELAY_32: 延迟 32 个时钟 EXT_START_DELAY_64: 延迟 64 个时钟 EXT_START_DELAY_128: 延迟 128 个时钟 EXT_START_DELAY_256: 延迟 256 个时钟 EXT_START_DELAY_512: 延迟 512 个时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17. 4. 27 函数 OSC_EXTLF_Start_Delay_Config

表 17-30 函数 OSC_EXTLF_Start_Delay_Config

函数名	OSC_EXTLF_Start_Delay_Config
函数原型	void OSC_EXTLF_Start_Delay_Config (uint32_t ExternalDelay)
功能描述	配置外部低频启动延迟控制
输入参数 1	ExternalDelay: 外部低频启动延迟控制, 取值范围为: EXT_START_DELAY_0: 延迟 0 个时钟 EXT_START_DELAY_2: 延迟 2 个时钟 EXT_START_DELAY_4: 延迟 4 个时钟 EXT_START_DELAY_8: 延迟 8 个时钟 EXT_START_DELAY_16: 延迟 16 个时钟 EXT_START_DELAY_32: 延迟 32 个时钟 EXT_START_DELAY_64: 延迟 64 个时钟 EXT_START_DELAY_128: 延迟 128 个时钟 EXT_START_DELAY_256: 延迟 256 个时钟 EXT_START_DELAY_512: 延迟 512 个时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17. 4. 28 函数 OSC_PLL_Software_Enable

表 17-31 函数 OSC_PLL_Software_Enable

函数名	OSC_PLL_Software_Enable
-----	-------------------------

函数原型	void OSC_PLL_Software_Enable (FunctionalState NewState)
功能描述	配置 PLL 软件使能
输入参数 1	NewState: PLL 软件使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.29 函数 OSC_EXTHF_Software_Enable

表 17-32 函数 OSC_EXTHF_Software_Enable

函数名	OSC_EXTHF_Software_Enable
函数原型	void OSC_EXTHF_Software_Enable (FunctionalState NewState)
功能描述	配置外部高频振荡器软件使能
输入参数 1	NewState: 外部高频振荡器软件使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.30 函数 OSC_EXTLF_Software_Enable

表 17-33 函数 OSC_EXTLF_Software_Enable

函数名	OSC_EXTLF_Software_Enable
函数原型	void OSC_EXTLF_Software_Enable (FunctionalState NewState)
功能描述	配置外部低频振荡器软件使能
输入参数 1	NewState: 外部低频振荡器软件使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.31 函数 OSC_INTHF_Software_Enable

表 17-34 函数 OSC_INTHF_Software_Enable

函数名	OSC_INTHF_Software_Enable
函数原型	void OSC_INTHF_Software_Enable (FunctionalState NewState)
功能描述	配置内部高频振荡器软件使能
输入参数 1	NewState: 内部高频振荡器软件使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.32 函数 OSC_INTLF_Software_Enable

表 17-35 函数 OSC_INTLF_Software_Enable

函数名	OSC_INTLF_Software_Enable
-----	---------------------------

函数原型	void OSC_INTLF_Software_Enable (FunctionalState NewState)
功能描述	配置内部低频振荡器软件使能
输入参数 1	NewState: 内部低频振荡器软件使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 33 函数 OSC_Zero_Drift_Config

表 17-36 函数 OSC_Zero_Drift_Config

函数名	OSC_Zero_Drift_Config
函数原型	void OSC_Zero_Drift_Config (uint32_t Scale, FunctionalState NewState)
功能描述	配置零温漂的绝对值校准权重位
输入参数 1	Scale: 零温漂电流档选择, 取值为下列宏的一个或多个组合: ZERO_DRIFT_SCALEDIV32: 1/32uA 电流档 ZERO_DRIFT_SCALEDIV16: 1/16uA 电流档 ZERO_DRIFT_SCALEDIV8: 1/8uA 电流档 ZERO_DRIFT_SCALEDIV4: 1/4uA 电流档 ZERO_DRIFT_SCALEDIV2: 1/2uA 电流档 ZERO_DRIFT_SCALE1X: 1uA 电流档 ZERO_DRIFT_SCALE2X: 2uA 电流档 ZERO_DRIFT_SCALE4X: 4uA 电流档 ZERO_DRIFT_SCALE8X: 8uA 电流档 ZERO_DRIFT_SCALE16X0: 16uA 电流档 ZERO_DRIFT_SCALE16X1: 16uA 电流档 ZERO_DRIFT_SCALE16X2: 16uA 电流档 ZERO_DRIFT_SCALE16X3: 16uA 电流档 ZERO_DRIFT_SCALE16X4: 16uA 电流档 ZERO_DRIFT_SCALE32X: 32uA 电流档
输入参数 2	NewState: 零温漂电流档状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 34 函数 OSC_Positive_Drift_Config

表 17-37 函数 OSC_Positive_Drift_Config

函数名	OSC_Positive_Drift_Config
函数原型	void OSC_Positive_Drift_Config (uint32_t PositiveDrift)
功能描述	配置内部高频振荡器的正温调节
输入参数 1	PositiveDrift: 内部高频振荡器的正温调节, 取值范围为: 0~31
返回值	无

被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)
-------	--

17.4.35 函数 OSC_Negative_Drift_Config

表 17-38 函数 OSC_Negative_Drift_Config

函数名	OSC_Negative_Drift_Config
函数原型	void OSC_Negative_Drift_Config (uint32_t PositiveDrift)
功能描述	配置内部高频振荡器的负温调节
输入参数 1	NegativeDrift: 内部高频振荡器的负温调节, 取值范围为: 0~31
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.36 函数 OSC_Current_Gain_Config

表 17-39 函数 OSC_Current_Gain_Config

函数名	OSC_Current_Gain_Config
函数原型	void OSC_Current_Gain_Config (uint32_t PositiveDrift)
功能描述	配置电流增益选择
输入参数 1	CurrentGain: 电流增益选择, 取值范围为: 0~3
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

17.4.37 函数 OSC_High_Speed_Enable

表 17-40 函数 OSC_High_Speed_Enable

函数名	OSC_High_Speed_Enable
函数原型	void OSC_High_Speed_Enable (FunctionalState NewState)
功能描述	配置高速选择
输入参数 1	NewState: 高速选择状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.38 函数 OSC_External_Input_Enable

表 17-41 函数 OSC_External_Input_Enable

函数名	OSC_External_Input_Enable
-----	---------------------------

函数原型	void OSC_External_Input_Enable (FunctionalState NewState)
功能描述	配置外部高频晶振输入使能
输入参数 1	NewState: 外部高频晶振输入使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 39 函数 OSC_Feedback_Resistance_Config

表 17-42 函数 OSC_Feedback_Resistance_Config

函数名	OSC_Feedback_Resistance_Config
函数原型	void OSC_Feedback_Resistance_Config (uint32_t NewState)
功能描述	配置反馈电阻调节
输入参数 1	NewState: 反馈电阻调节, 取值范围为: FREQUENCY_MORE_THAN_10M: 频率大于 10MHz FREQUENCY_LESS_THAN_10M: 频率小于 10MHz
返回值	无
被调用函数	无

17. 4. 40 函数 OSC_PLL_Zero_Source_Enable

表 17-43 函数 OSC_PLL_Zero_Source_Enable

函数名	OSC_PLL_Zero_Source_Enable
函数原型	void OSC_PLL_Zero_Source_Enable (FunctionalState NewState)
功能描述	配置 PLL 零温漂电流源输出到 IO 口使能
输入参数 1	NewState: PLL 零温漂电流源输出到 IO 口使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 41 函数 OSC_PLL_Vref2_Enable

表 17-44 函数 OSC_PLL_Vref2_Enable

函数名	OSC_PLL_Vref2_Enable
函数原型	void OSC_PLL_Vref2_Enable (FunctionalState NewState)
功能描述	配置 PLL 内部参考电压 2V 输出到 IO 口的使能
输入参数 1	NewState: PLL 内部参考电压 2V 输出到 IO 口的使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.42 函数 OSC_PLL_Vref0p5_Enable

表 17-45 函数 OSC_PLL_Vref0p5_Enable

函数名	OSC_PLL_Vref0p5_Enable
函数原型	void OSC_PLL_Vref0p5_Enable (FunctionalState NewState)
功能描述	配置 PLL 内部低功耗 BG 参考 0.5V 到内部 buffer 的使能
输入参数 1	NewState: PLL 内部低功耗 BG 参考 0.5V 到内部 buffer 的使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.43 函数 OSC_PLL_Vref1p2_Enable

表 17-46 函数 OSC_PLL_Vref1p2_Enable

函数名	OSC_PLL_Vref1p2_Enable
函数原型	void OSC_PLL_Vref1p2_Enable (FunctionalState NewState)
功能描述	配置 PLL 主 BG 到内部 buffer 的通道使能
输入参数 1	NewState: PLL 主 BG 到内部 buffer 的通道使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.44 函数 OSC_PLL_Low_Power_20nA_Enable

表 17-47 函数 OSC_PLL_Low_Power_20nA_Enable

函数名	OSC_PLL_Low_Power_20nA_Enable
函数原型	void OSC_PLL_Low_Power_20nA_Enable (FunctionalState NewState)
功能描述	配置 PLL 内部低功耗 BG 的电流源 20nA 到 IO 口的选通使能
输入参数 1	NewState: PLL 内部低功耗 BG 的电流源 20nA 到 IO 口的选通使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.45 函数 OSC_PLL_Vref1p14_Enable

表 17-48 函数 OSC_PLL_Vref1p14_Enable

函数名	OSC_PLL_Vref1p14_Enable
函数原型	void OSC_PLL_Vref1p14_Enable (FunctionalState NewState)
功能描述	配置 PLL 内部低功耗参考电压 1.14V 到内部 buffer 的选通使能
输入参数 1	NewState: PLL 内部低功耗参考电压 1.14V 到内部 buffer 的选通使能状态,

	取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 46 函数 OSC_PLL_Low_Power_100nA_Enable

表 17-49 函数 OSC_PLL_Low_Power_100nA_Enable

函数名	OSC_PLL_Low_Power_100nA_Enable
函数原型	void OSC_PLL_Low_Power_100nA_Enable (FunctionalState NewState)
功能描述	配置 PLL 内部低功耗偏置电路的电流源 100nA 到 IO 口的选通使能
输入参数 1	NewState: PLL 内部低功耗偏置电路的电流源 100nA 到 IO 口的选通使能状态，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 47 函数 OSC_PLL_LDO_Enable

表 17-50 函数 OSC_PLL_LDO_Enable

函数名	OSC_PLL_LDO_Enable
函数原型	void OSC_PLL_LDO_Enable (FunctionalState NewState)
功能描述	配置 PLL 内部高频 LDO 到内部 buffer 的选通使能
输入参数 1	NewState: PLL 内部高频 LDO 到内部 buffer 的选通使能状态，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

17. 4. 48 函数 OSC_PLL_INT_Enable

表 17-51 函数 OSC_PLL_INT_Enable

函数名	OSC_PLL_INT_Enable
函数原型	void OSC_PLL_INT_Enable (FunctionalState NewState)
功能描述	配置 PLL 中断使能
输入参数 1	NewState: PLL 中断使能状态，取值范围为：TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.49 函数 OSC_EXTHF_INT_Enable

表 17-52 函数 OSC_EXTHF_INT_Enable

函数名	OSC_EXTHF_INT_Enable
函数原型	void OSC_EXTHF_INT_Enable (FunctionalState NewState)
功能描述	配置外部高频中断使能
输入参数 1	NewState: 外部高频中断使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.50 函数 OSC_EXTLF_INT_Enable

表 17-53 函数 OSC_EXTLF_INT_Enable

函数名	OSC_EXTLF_INT_Enable
函数原型	void OSC_EXTLF_INT_Enable (FunctionalState NewState)
功能描述	配置外部低频中断使能
输入参数 1	NewState: 外部低频中断使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.51 函数 OSC_INTHF_INT_Enable

表 17-54 函数 OSC_INTHF_INT_Enable

函数名	OSC_INTHF_INT_Enable
函数原型	void OSC_INTHF_INT_Enable (FunctionalState NewState)
功能描述	配置内部高频中断使能
输入参数 1	NewState: 内部高频中断使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.52 函数 OSC_INTLF_INT_Enable

表 17-55 函数 OSC_INTLF_INT_Enable

函数名	OSC_INTLF_INT_Enable
函数原型	void OSC_INTLF_INT_Enable (FunctionalState NewState)
功能描述	配置内部低频中断使能
输入参数 1	NewState: 内部低频中断使能状态, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

17.4.53 函数 OSC_Get_Clock_Failure_INT_Flag

表 17-56 函数 OSC_Get_Clock_Failure_INT_Flag

函数名	OSC_Get_Clock_Failure_INT_Flag
函数原型	FlagStatus OSC_Get_Clock_Failure_INT_Flag (void)
功能描述	读取时钟故障标志
输入参数 1	无
返回值	时钟故障状态，0：没有时钟故障，1：检测到时钟故障
被调用函数	无

17.4.54 函数 OSC_Get_PLL_INT_Flag

表 17-57 函数 OSC_Get_PLL_INT_Flag

函数名	OSC_Get_PLL_INT_Flag
函数原型	FlagStatus OSC_Get_PLL_INT_Flag (void)
功能描述	读取 PLL 中断标志
输入参数 1	无
返回值	中断状态，0：未发生中断，1：发生中断
被调用函数	无

17.4.55 函数 OSC_Get_EXTHF_INT_Flag

表 17-58 函数 OSC_Get_EXTHF_INT_Flag

函数名	OSC_Get_EXTHF_INT_Flag
函数原型	FlagStatus OSC_Get_EXTHF_INT_Flag (void)
功能描述	读取外部高频中断标志
输入参数 1	无
返回值	中断状态，0：未发生中断，1：发生中断
被调用函数	无

17.4.56 函数 OSC_Get_EXTLF_INT_Flag

表 17-59 函数 OSC_Get_EXTLF_INT_Flag

函数名	OSC_Get_EXTLF_INT_Flag
函数原型	FlagStatus OSC_Get_EXTLF_INT_Flag (void)
功能描述	读取外部低频中断标志
输入参数 1	无
返回值	中断状态，0：未发生中断，1：发生中断
被调用函数	无

17.4.57 函数 OSC_Get_INTHF_INT_Flag

表 17-60 函数 OSC_Get_INTHF_INT_Flag

函数名	OSC_Get_INTHF_INT_Flag
函数原型	FlagStatus OSC_Get_INTHF_INT_Flag (void)
功能描述	读取内部高频中断标志
输入参数 1	无
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

17.4.58 函数 OSC_Get_INTLF_INT_Flag

表 17-61 函数 OSC_Get_INTLF_INT_Flag

函数名	OSC_Get_INTLF_INT_Flag
函数原型	FlagStatus OSC_Get_INTLF_INT_Flag (void)
功能描述	读取内部低频中断标志
输入参数 1	无
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

17.4.59 函数 OSC_Get_LP4MIF_INT_Flag

表 17-62 函数 OSC_Get_LP4MIF_INT_Flag

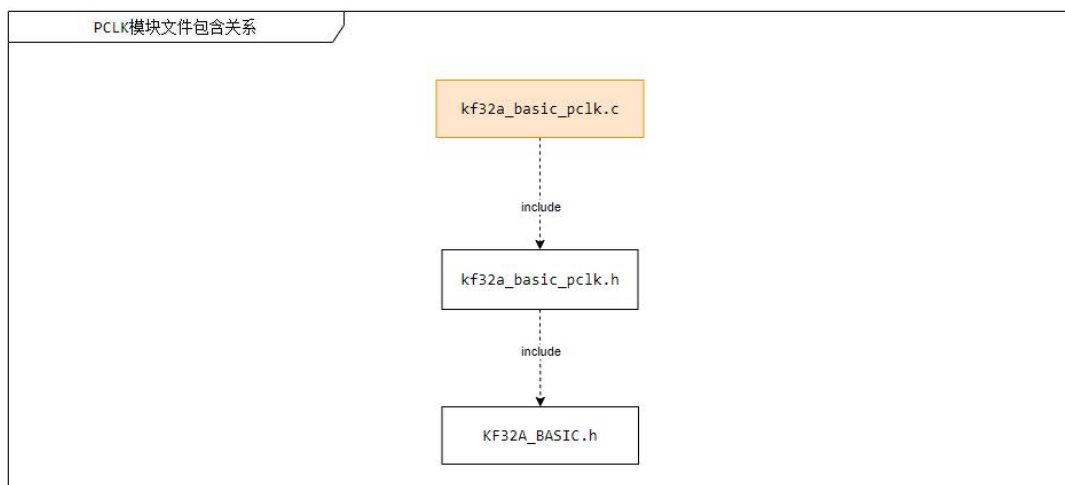
函数名	OSC_Get_LP4MIF_INT_Flag
函数原型	FlagStatus OSC_Get_LP4MIF_INT_Flag (void)
功能描述	读取 LP4M 中断标志
输入参数 1	无
返回值	中断状态, 0: 未发生中断, 1: 发生中断
被调用函数	无

18 外设模块时钟使能模块（PCLK）

为了降低功耗，默认外设时钟就禁止。在使用外设模块时，需要使能该外设模块时钟控制信号，否则模块不工作。通过 PCLK_CTLx（x=0,1,2,3）外设时钟控制寄存器控制相应的外设时钟。当外设时钟禁止时，CPU 无法对相应的模块寄存器进行写操作。

18.1 文件引用关系

图 18-1 PCLK 模块文件包含关系



18.2 PCLK 寄存器结构

PCLK 寄存器结构，PCLK_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct PCLK_MemMap {
    volatile uint32_t    CTL0;
    volatile uint32_t    CTL1;
    volatile uint32_t    CTL2;
    volatile uint32_t    CTL3;
}PCLK_SFRmap;
  
```

表 18-1 PCLK 寄存器结构说明

寄存器	描述
CTL0	PCLK 控制寄存器 0，偏移:0x00
CTL1	PCLK 控制寄存器 1，偏移:0x04
CTL2	PCLK 控制寄存器 2，偏移:0x08
CTL3	PCLK 控制寄存器 3，偏移:0x0C

18.3 PCLK 宏定义

PCLK 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, PCLK 其他相关宏定义详见文件 kf32a_basic_pclk.h 及函数参数描述。

18.4 PCLK 库函数

表 18-2 PCLK 固件库函数列表

序号	函数名	描述
1	PCLK_CTL0_Peripheral_Clock_Enable	配置 PCLK_CTL0 控制的外设时钟使能
2	PCLK_CTL1_Peripheral_Clock_Enable	配置 PCLK_CTL1 控制的外设时钟使能
3	PCLK_CTL2_Peripheral_Clock_Enable	配置 PCLK_CTL2 控制的外设时钟使能
4	PCLK_CTL3_Peripheral_Clock_Enable	配置 PCLK_CTL3 控制的外设时钟使能

18.4.1 函数 PCLK_CTL0_Peripheral_Clock_Enable

表 18-3 函数 PCLK_CTL0_Peripheral_Clock_Enable

函数名	PCLK_CTL0_Peripheral_Clock_Enable
函数原型	void PCLK_CTL0_Peripheral_Clock_Enable (uint32_t PCLK_CTL0_bit,FunctionalState NewState)
功能描述	配置 PCLK_CTL0 控制的外设时钟使能
输入参数 1	PCLK_CTL0_bit: 指定外设时钟使能位的掩码, 取值为: PCLK_CTL0_GPIOACLKEN: GPIOA PCLK_CTL0_GPIOBCLKEN: GPIOB PCLK_CTL0_GPIOCCLKEN: GPIOC PCLK_CTL0_GPIODCLKEN: GPIOD PCLK_CTL0_GPIOECLKEN: GPIOE PCLK_CTL0_GPIOFCLKEN: GPIOF PCLK_CTL0_GPIOGCLKEN: GPIOG PCLK_CTL0_GPIOHCLKEN: GPIOH
输入参数 2	NewState: 外设时钟使能状态, 取值范围为 TRUE 或 FALSE
返回值	无
被调用函数	无

18.4.2 函数 PCLK_CTL1_Peripheral_Clock_Enable

表 18-4 函数 PCLK_CTL1_Peripheral_Clock_Enable

函数名	PCLK_CTL1_Peripheral_Clock_Enable
函数原型	void PCLK_CTL1_Peripheral_Clock_Enable (uint32_t PCLK_CTL1_bit,FunctionalState NewState)
功能描述	配置 PCLK_CTL1 控制的外设时钟使能

输入参数 1	PCLK_CTL1_bit: 指定外设时钟使能位的掩码，取值为： PCLK_CTL1_QEI1CLKEN: QEI1 PCLK_CTL1_T1CLKEN: T1 PCLK_CTL1_T2CLKEN: T2 PCLK_CTL1_T3CLKEN: T3 PCLK_CTL1_T4CLKEN: T4 PCLK_CTL1_T5T6CLKEN: T5 PCLK_CTL1_T9T10CLKEN: T9 PCLK_CTL1_QEI0CLKEN: QEI0 PCLK_CTL1_ADC0CLKEN: ADC0 PCLK_CTL1_ADC1CLKEN: ADC1 PCLK_CTL1_ADC2CLKEN: ADC2 PCLK_CTL1_DAC0CLKEN: DAC0 PCLK_CTL1_DAC1CLKEN: DAC1 PCLK_CTL1_CMPCLKEN: CMP PCLK_CTL1_T0CLKEN: T0 PCLK_CTL1_CTOUCHCLKEN: CTOUCH PCLK_CTL1_USART0CLKEN: USART0 PCLK_CTL1_USART1CLKEN: USART1 PCLK_CTL1_USART2CLKEN: USART2 PCLK_CTL1_USART3CLKEN: USART3 PCLK_CTL1_USART4CLKEN: USART4 PCLK_CTL1_SPI0CLKEN: SPI0 PCLK_CTL1_SPI1CLKEN: SPI1 PCLK_CTL1_I2C0CLKEN: I2C0 PCLK_CTL1_I2C1CLKEN: I2C1 PCLK_CTL1_I2C2CLKEN: I2C2 PCLK_CTL1_LCDCLKEN: LCD PCLK_CTL1_USBCLKEN: USB
输入参数 2	NewState: 外设时钟使能状态，取值范围为 TRUE 或 FALSE
返回值	无
被调用函数	无

18.4.3 函数 PCLK_CTL2_Peripheral_Clock_Enable

表 18-5 函数 PCLK_CTL2_Peripheral_Clock_Enable

函数名	PCLK_CTL2_Peripheral_Clock_Enable
函数原型	void PCLK_CTL2_Peripheral_Clock_Enable (uint32_t PCLK_CTL2_bit, FunctionalState NewState)
功能描述	配置 PCLK_CTL2 控制的外设时钟使能
输入参数 1	PCLK_CTL2_bit: 指定外设时钟使能位的掩码，取值为：

	PCLK_CTL2_CAN0CLKEN: CAN0 PCLK_CTL2_CAN1CLKEN: CAN1 PCLK_CTL2_WWDTCLKEN: WWDT PCLK_CTL2_DMA0CLKEN: DMA0 PCLK_CTL2_DMA1CLKEN: DMA1 PCLK_CTL2_T14CLKEN: T14 PCLK_CTL2_T15CLKEN: T15 PCLK_CTL2_CAN2CLKEN: CAN2 PCLK_CTL2_CAN3CLKEN: CAN3 PCLK_CTL2_T18CLKEN: T18 PCLK_CTL2_T19CLKEN: T19 PCLK_CTL2_T20CLKEN: T20 PCLK_CTL2_T21CLKEN: T21 PCLK_CTL2_T21CLKEN: T22 PCLK_CTL2_T21CLKEN: T23 PCLK_CTL2_USART5CLKEN: USART5 PCLK_CTL2_USART6CLKEN: USART6 PCLK_CTL2_USART7CLKEN: USART7
输入参数 2	NewState: 外设时钟使能状态, 取值范围为 TRUE 或 FALSE
返回值	无
被调用函数	无

18.4.4 函数 PCLK_CTL3_Peripheral_Clock_Enable

表 18-6 函数 PCLK_CTL3_Peripheral_Clock_Enable

函数名	PCLK_CTL3_Peripheral_Clock_Enable
函数原型	void PCLK_CTL3_Peripheral_Clock_Enable (uint32_t PCLK_CTL3_bit, FunctionalState NewState)
功能描述	配置 PCLK_CTL3 控制的外设时钟使能
输入参数 1	PCLK_CTL3_bit: 指定外设时钟使能位的掩码, 取值为: PCLK_CTL3_SPI2CLKEN: SPI2 PCLK_CTL3_SPI3CLKEN: SPI3 PCLK_CTL3_I2C3CLKEN: I2C3 PCLK_CTL3_CRCCLKEN: CRC PCLK_CTL3_AESCLKEN: AES PCLK_CTL3_LEDCLKEN: LED PCLK_CTL3_EXICCLKEN: EXIC PCLK_CTL3_CAN4CLKEN: CAN4 PCLK_CTL3_CAN5CLKEN: CAN5 PCLK_CTL3_CFGLCLKEN: CFGL PCLK_CTL3_OPCLKEN: OP

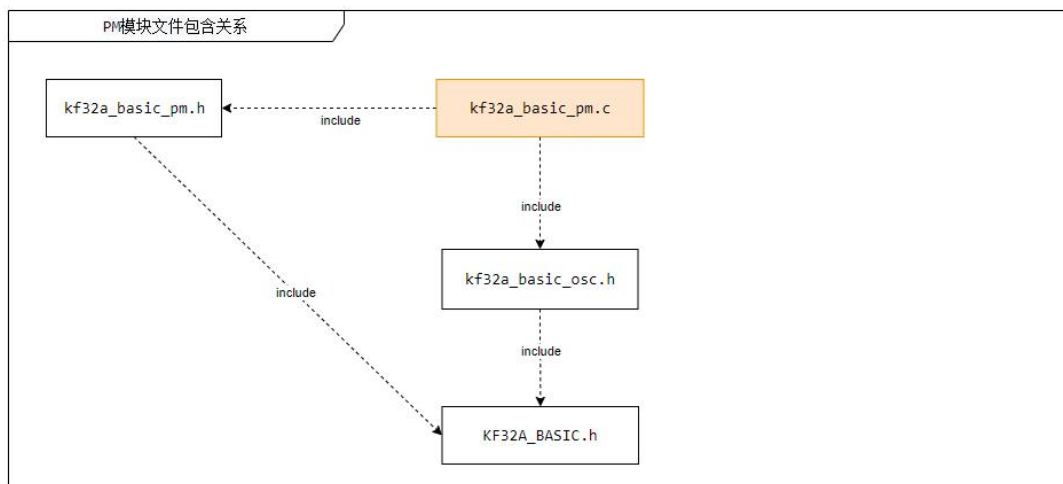
返回值	无
被调用函数	无

19 电源管理（PM）

电源管理模块（PM）是控制 MCU 运行在不同工作模式的模块。MCU 可运行在：正常运行模式、休眠模式、低功耗模式。

19.1 文件引用关系

图 19-1 PM 模块文件包含关系



19.2 PM 寄存器结构

PM 寄存器结构，PM_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct PM_MemMap {
    volatile uint32_t    CTL0;
    volatile uint32_t    CTL1;
    volatile const uint32_t    STA0;
    volatile const uint32_t    STA1;
    volatile uint32_t    STAC;
    volatile uint32_t    CTL2;
    volatile uint32_t    CAL0;
    volatile uint32_t    CAL1;
    volatile uint32_t    CAL2;
}PM_SFRmap;
  
```

表 19-1 PM 寄存器结构说明

寄存器	描述
CTL0	PM 功耗模式控制寄存器 0，偏移;0x00
CTL1	PM 功耗模式控制寄存器 1，偏移;0x04

STA0	PM 功耗模式状态寄存器 0, 偏移;0x08
STA1	PM 功耗模式状态寄存器 1, 偏移;0x0C
STAC	PM 功耗模式状态清零寄存器, 偏移;0x10
CTL2	PM 功耗模式控制寄存器 2, 偏移;0x14
CAL0	PM 校准寄存器 0, 偏移;0X18
CAL1	PM 校准寄存器 1, 偏移;0X1C
CAL2	PM 校准寄存器 1, 偏移;0X20

19.3 PM 宏定义

PM 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, PM 其他相关宏定义详见文件 kf32a_basic_pm.h 及函数参数描述。

19.4 PM 库函数

表 19-2 PM 固件库函数列表

序号	函数名	描述
1	PM_IO_Latch_Enable	设置 IO 口状态锁存使能
2	PM_Get_IO_Latch_Status	获取 IO 口状态锁存
3	PM_Internal_Low_Frequency_Enable	设置内部低频振荡器软件使能
4	PM_External_Low_Frequency_Enable	设置外部低频振荡器软件使能
5	PM_External_Low_Frequency_Clock_Enable	设置外部低频振荡器时钟输入使能
6	PM_Main_Bandgap_Enable	设置主 BG 软件使能
7	PM_LDO18_Enable	设置 LDO18 软件使能
8	PM_Backup_Registers_Reset_Config	设置备份区寄存器模块软件复位
9	PM_Independent_Watchdog_Reset_Config	设置 IWDG 模块软件复位
10	PM_SRAMA_In_Standby_Work_Mode_Config	设置 SRAM 的 A 区在 standby 模式下工作配置
11	PM_LPRAM_In_Standby_Work_Mode_Config	设置 LPRAM 在 standby 模式下工作配置
12	PM_Backup_POR_Delay_Time_Config	设置 BKP_POR 延时时间
13	PM_Main_POR_Delay_Time_Config	设置主 POR、POR12 和 POR18 延时时间
14	PM_Peripheral_IO_Port_Config	设置低功耗外设 IO 口
15	PM_OCAL0LOCK_Enable	低功耗模式下锁存内部高频晶振校准
16	PM_MEMSEL_Enable	低功耗模式下 MEMM 数据保持使
17	PM_Flash_Power_Off_Enable	设置 FLASH 供电电源软件关断使能
18	PM_CCP0CLKLPEN_Enable	设置 CCP0 内部低频时钟使能
19	PM_Backup_Write_And_Read_Enable	设置备份区读写操作允许使能

20	PM_VREF_Software_Enable	设置参考电压使能
21	PM_VREF_SELECT	设置参考电压档位选择
22	PM_LPR_Software_Enable	设置低功耗电压调制器 LPR 软件使能
23	PM_Low_Power_Mode_Config	设置低功耗模式
24	PM_BOR_Enable	设置 BOR 使能
25	PM_Low_Power_BOR_Enable	设置 LPBOR 使能
26	PM_Temperature_Sensor_Enable	设置 TEMPSSENSOR 使能
27	PM_Temperature_Sensor_Buffer_Enable	设置 TEMPSSENSOR 输出到 BUFFER 使能
28	PM_Reference_Voltage_Enable	设置参考电压 2V 模块使能
29	PM_Internal_Test_Buffer_Clock_Enable	设置内部测试输出 BUFFER 时钟使能
30	PM_Internal_Test_Buffer_Clock_Scaler_Config	设置内部测试输出 BUFFER 工作时钟分频
31	PM_PLL0_Output_Buffer_Enable	设置 PLL0 时钟输出到 BUFFER 使能
32	PM_PLL1_Output_Buffer_Enable	设置 PLL1 时钟输出到 BUFFER 使能
33	PM_PLL2_Output_Buffer_Enable	设置 PLL2 时钟输出到 BUFFER 使能
34	PM_PLL0LDO_Output_Buffer_Enable	设置 PLL0_LDO 输出到 BUFFER 使能
35	PM_PLL1LDO_Output_Buffer_Enable	设置 PLL1_LDO 输出到 BUFFER 使能
36	PM_PLL2LDO_Output_Buffer_Enable	设置 PLL2_LDO 输出到 BUFFER 使能
37	PM_Battery_BOR_Config	设置 BAT_BOR 电压点选择
38	PM_Battery_BOR_Enable	设置 BAT_BOR 使能
39	PM_Peripheral_Voltage_Monitoring_Enable	设置 PVM1 使能
40	PM_Voltage_Detection_Config	设置电压检测点选择
41	PM_Voltage_Detection_Enable	设置电压检测功能使能
42	PM_External_Wakeup_Pin_Enable	设置外部唤醒引脚 WKPx 使能
43	PM_External_Wakeup_Edge_Config	设置外部唤醒引脚 WKPx 唤醒触发沿
44	PM_Stop_Mode_Peripheral_INLF_Enable	设置不同外设工作在 Stop 模式下内部低频时钟使能
45	PM_Peripheral_Reset_Config	设置不同外设复位
46	PM_Vdd_Por_Enable	设置主电源 POR 强制使能
47	PM_Low_Power_Bandgap_Enable	设置 LP_BG 软件使能
48	PM_Power_Dissipation_Mode_Config	设置功耗模式时序选择

49	PM_Power_Dissipation_Mode_Delay_Config	设置 POR12 关闭到 LDO 关闭的延时时间
50	PM_Internal_Test_Buffer_Enable	设置内部测试输出 BUFFER 使能
51	PM_Clear_Reset_And_Wakeup_Flag	清零复位/唤醒状态标志位
52	PM_Get_IWDT_Reset_Flag	获取对应复位/唤醒事件的状态标志
53	PM_Clear_External_Wakeup_Pin_Flag	清零外部唤醒引脚 WKPx 状态标志
54	PM_Get_Low_Power_Running_State	获取低功耗运行模式状态标志
55	PM_Get_LPR_Status	获取低功耗电压调制器 LPR 状态
56	PM_Get_Peripheral_Voltage_Detection_Status	获取电压检测状态
57	PM_Zero_Drift_Current_Config	设置零温漂电流档校准
58	PM_BOR_Voltage_Config	设置 BOR 电压点选择
59	PM_Main_Regulator_Voltage_Config	设置 MR 或 MR_HV 模块校准
60	PM_Main_Regulator_HV_Enable	设置 MR_HV 模块软件使能
61	PM_Reference_Calibration_Config	设置不同的基准校准
62	PM_INTLF_Bias_Current_Config	设置内部低频振荡器电流校准
63	PM_EXTLF_Bias_Current_Config	设置外部低频振荡器电流校准
64	PM_INTLF_Capacitance_Calibration_Config	设置内部低频振荡器电容校准
65	PM_LP_Bias_Calibration_Config	设置低功耗偏置电流校准
66	PM_LPBG_Pump_Calibration_Config	设置内部低频振荡器电容校准
67	PM_EXTLF_N_Bias_Current_Config	设置外部低频振荡器 N 管偏置校准
68	PM_EXTLF_PIN_Selection_Config	设置外部低频振荡器引脚选择位
69	PM_EXTHF_PIN_Selection_Config	设置外部高频振荡器引脚选择位
70	PM_LDO18_Module_Config	设置 LDO18 内置模块选择
71	PM_Main_Regulator_Bandgap_Config	设置 MR_BG 模式选择
72	PM_LPR_Module_Config	设置 LPR 内置模块选择

19.4.1 函数 PM_IO_Latch_Enable

表 19-3 函数 PM_IO_Latch_Enable

函数名	PM_IO_Latch_Enable
函数原型	void PM_IO_Latch_Enable (FunctionalState NewState)
功能描述	设置 IO 口状态锁存使能
输入参数 1	NewState: IO 口状态锁存使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.2 函数 PM_Get_IO_Latch_Status

表 19-4 函数 PM_Get_IO_Latch_Status

函数名	PM_Get_IO_Latch_Status
-----	------------------------

函数原型	FlagStatus PM_Get_IO_Latch_Status (void)
功能描述	获取 IO 口状态锁存
输入参数 1	无
返回值	1:IO 口状态被锁存；0:IO 口状态未被锁存
被调用函数	无

19.4.3 函数 PM_Internal_Low_Frequency_Enable

表 19-5 函数 PM_Internal_Low_Frequency_Enable

函数名	PM_Internal_Low_Frequency_Enable
函数原型	void PM_Internal_Low_Frequency_Enable (FunctionalState NewState)
功能描述	设置内部低频振荡器软件使能
输入参数 1	NewState: 内部低频振荡器软件使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.4 函数 PM_External_Low_Frequency_Enable

表 19-6 函数 PM_External_Low_Frequency_Enable

函数名	PM_External_Low_Frequency_Enable
函数原型	void PM_External_Low_Frequency_Enable (FunctionalState NewState)
功能描述	设置外部低频振荡器软件使能
输入参数 1	NewState: 外部低频振荡器软件使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.5 函数 PM_External_Low_Frequency_Clock_Enable

表 19-7 函数 PM_External_Low_Frequency_Clock_Enable

函数名	PM_External_Low_Frequency_Clock_Enable
函数原型	void PM_External_Low_Frequency_Clock_Enable (FunctionalState NewState)
功能描述	设置外部低频振荡器时钟输入使能
输入参数 1	NewState: 外部低频振荡器时钟输入使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.6 函数 PM_Main_Bandgap_Enable

表 19-8 函数 PM_Main_Bandgap_Enable

函数名	PM_Main_Bandgap_Enable
函数原型	void PM_Main_Bandgap_Enable (FunctionalState NewState)
功能描述	设置主 BG 软件使能
输入参数 1	NewState: 主 BG 软件使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.7 函数 PM_LDO18_Enable

表 19-9 函数 PM_LDO18_Enable

函数名	PM_LDO18_Enable
函数原型	void PM_LDO18_Enable (FunctionalState NewState)
功能描述	设置 LDO18 软件使能
输入参数 1	NewState: LDO18 软件使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.8 函数 PM_Backup_Registers_Reset_Config

表 19-10 函数 PM_Backup_Registers_Reset_Config

函数名	PM_Backup_Registers_Reset_Config
函数原型	void PM_Backup_Registers_Reset_Config (uint32_t BkpReset)
功能描述	设置备份区寄存器模块软件复位
输入参数 1	BkpReset: 备份区寄存器模块软件复位状态, 取值为: PERIPHERAL_RST_STATUS: 备份区寄存器模块处于复位状态 PERIPHERAL_OUTRST_STATUS: 备份区寄存器模块退出复位
返回值	无
被调用函数	无

19.4.9 函数 PM_Independent_Watchdog_Reset_Config

表 19-11 函数 PM_Independent_Watchdog_Reset_Config

函数名	PM_Independent_Watchdog_Reset_Config
函数原型	void PM_Independent_Watchdog_Reset_Config (uint32_t IWDTRReset)
功能描述	设置 IWDTR 模块软件复位
输入参数 1	IWDTRReset: IWDTR 模块软件复位状态, 取值为: PERIPHERAL_RST_STATUS: IWDTR 处于复位状态

	PERIPHERAL_OUTRST_STATUS: IWDG 退出复位
返回值	无
被调用函数	无

19.4.10 函数 PM_SRAMA_In_Standby_Work_Mode_Config

表 19-12 函数 PM_SRAMA_In_Standby_Work_Mode_Config

函数名	PM_SRAMA_In_Standby_Work_Mode_Config
函数原型	void PM_SRAMA_In_Standby_Work_Mode_Config (uint32_t WorkMode)
功能描述	设置 SRAM 的 A 区在 standby 模式下工作配置
输入参数 1	WorkMode: SRAM 的 A 区在 standby 模式下工作配置, 取值为: PM_SRAMA_IN_STANDBY_POWER_DOWN: SRAM 的 A 区在 standby 模式下掉电 PM_SRAMA_IN_STANDBY_KEEP_DATA: SRAM 的 A 区在 standby 模式下保持数据
返回值	无
被调用函数	无

19.4.11 函数 PM_LPRAM_In_Standby_Work_Mode_Config

表 19-13 函数 PM_LPRAM_In_Standby_Work_Mode_Config

函数名	PM_LPRAM_In_Standby_Work_Mode_Config
函数原型	void PM_LPRAM_In_Standby_Work_Mode_Config (uint32_t WorkMode)
功能描述	设置 LPRAM 在 standby 模式下工作配置
输入参数 1	WorkMode: LPRAM 在 standby 模式下工作配置, 取值为: PM_LPRAM_IN_STANDBY_POWER_DOWN: LPRAM 在 standby 模式下掉电 PM_LPRAM_IN_STANDBY_KEEP_DATA: LPRAM 在 standby 模式下保持数据
返回值	无
被调用函数	无

19.4.12 函数 PM_Backup_POR_Delay_Time_Config

表 19-14 函数 PM_Backup_POR_Delay_Time_Config

函数名	PM_Backup_POR_Delay_Time_Config
函数原型	void PM_Backup_POR_Delay_Time_Config (uint32_t DelayTime)
功能描述	设置 BKP_POR 延时时间
输入参数 1	DelayTime: BKP_POR 延时时间, 取值为: DELAY_TIME_1ms: 1ms

	DELAY_TIME_32us: 32us
返回值	无
被调用函数	无

19.4.13 函数 PM_Main_POR_Delay_Time_Config

表 19-15 函数 PM_Main_POR_Delay_Time_Config

函数名	PM_Main_POR_Delay_Time_Config
函数原型	void PM_Main_POR_Delay_Time_Config (uint32_t DelayTime)
功能描述	设置主 POR、POR12 和 POR18 延时时间
输入参数 1	DelayTime: 主 POR、POR12 和 POR18 延时时间, 取值为: DELAY_TIME_2ms: 2ms DELAY_TIME_32us: 32us
返回值	无
被调用函数	无

19.4.14 函数 PM_Peripheral_IO_Port_Config

表 19-16 函数 PM_Peripheral_IO_Port_Config

函数名	PM_Peripheral_IO_Port_Config
函数原型	void PM_Peripheral_IO_Port_Config (uint32_t PeripheralPort)
功能描述	设置低功耗外设 IO 口
输入参数 1	PeripheralPort: 低功耗外设 IO 口, 取值为: PM_GENERAL_PURPOSE_IO_PORT: 通用 IO 口 PM_LOW_POWER_IO_PORT: 低功耗专用 IO 口
返回值	无
被调用函数	无

19.4.15 函数 PM_OCAL0LOCK_Enable

表 19-17 函数 PM_OCAL0LOCK_Enable

函数名	PM_OCAL0LOCK_Enable
函数原型	void PM_OCAL0LOCK_Enable (FunctionalState NewState)
功能描述	低功耗模式下锁存内部高频晶振校准
输入参数 1	NewState: HSI 内部高频晶振校准值锁存位, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.16 函数 PM_MEMSEL_Enable

表 19-18 函数 PM_MEMSEL_Enable

函数名	PM_MEMSEL_Enable
函数原型	void PM_MEMSEL_Enable (FunctionalState NewState)
功能描述	低功耗模式下 MEMM 数据保持使
输入参数 1	NewState: MEMM 数据保持使能位,取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.17 函数 PM_Flash_Power_Off_Enable

表 19-19 函数 PM_Flash_Power_Off_Enable

函数名	PM_Flash_Power_Off_Enable
函数原型	void PM_Flash_Power_Off_Enable (FunctionalState NewState)
功能描述	设置 FLASH 供电电源软件关断使能
输入参数 1	NewState: FLASH 供电电源软件关断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.18 函数 PM_CCP0CLKLPEN_Enable

表 19-20 函数 PM_CCP0CLKLPEN_Enable

函数名	PM_CCP0CLKLPEN_Enable
函数原型	void PM_CCP0CLKLPEN_Enable (FunctionalState NewState)
功能描述	设置 CCP0 内部低频时钟使能
输入参数 1	NewState: CCP0 内部低频时钟使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.19 函数 PM_Backup_Write_And_Read_Enable

表 19-21 函数 PM_Backup_Write_And_Read_Enable

函数名	PM_Backup_Write_And_Read_Enable
函数原型	void PM_Backup_Write_And_Read_Enable (FunctionalState NewState)
功能描述	设置备份区读写操作允许使能
输入参数 1	NewState: 备份区读写操作允许使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.20 函数 PM_VREF_Software_Enable

表 19-22 函数 PM_VREF_Software_Enable

函数名	PM_VREF_Software_Enable
函数原型	void PM_VREF_Software_Enable (FunctionalState NewState)
功能描述	设置参考电压使能
输入参数 1	NewState: 参考电压使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.21 函数 PM_VREF_SELECT

表 19-23 函数 PM_VREF_SELECT

函数名	PM_VREF_SELECT
函数原型	void PM_VREF_SELECT (uint32_t Voltage)
功能描述	设置参考电压档位选择
输入参数 1	NewState: 参考电压档位选择: PM_VREF_VOLTAGE_2P0V PM_VREF_VOLTAGE_1P5V PM_VREF_VOLTAGE_2P5V PM_VREF_VOLTAGE_3P0V
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.22 函数 PM_LPR_Software_Enable

表 19-24 函数 PM_LPR_Software_Enable

函数名	PM_LPR_Software_Enable
函数原型	void PM_LPR_Software_Enable (FunctionalState NewState)
功能描述	设置低功耗电压调制器 LPR 软件使能
输入参数 1	NewState: 低功耗电压调制器 LPR 软件使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.23 函数 PM_Low_Power_Mode_Config

表 19-25 函数 PM_Low_Power_Mode_Config

函数名	PM_Low_Power_Mode_Config
函数原型	void PM_Low_Power_Mode_Config (uint32_t LowPowerMode)

功能描述	设置低功耗模式
输入参数 1	LowPowerMode: 低功耗模式, 取值为: PM_LOW_POWER_MODE_STOP_0: 停止模式 0 PM_LOW_POWER_MODE_STOP_1: 停止模式 1 PM_LOW_POWER_MODE_STANDBY: 待机模式 PM_LOW_POWER_MODE_SHUTDOWN: 关断模式
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.24 函数 PM_BOR_Enable

表 19-26 函数 PM_BOR_Enable

函数名	PM_BOR_Enable
函数原型	void PM_BOR_Enable (FunctionalState NewState)
功能描述	设置 LPBOR 使能
输入参数 1	NewState: BOR 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.25 函数 PM_Low_Power_BOR_Enable

表 19-27 函数 PM_Low_Power_BOR_Enable

函数名	PM_Low_Power_BOR_Enable
函数原型	void PM_Low_Power_BOR_Enable (FunctionalState NewState)
功能描述	设置 TEMPSSENSOR 使能
输入参数 1	NewState: LPBOR 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.26 函数 PM_Temperature_Sensor_Enable

表 19-28 函数 PM_Temperature_Sensor_Enable

函数名	PM_Temperature_Sensor_Enable
函数原型	void PM_Temperature_Sensor_Enable (FunctionalState NewState)
功能描述	设置 TEMPSSENSOR 使能
输入参数 1	NewState: TEMPSSENSOR 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.27 函数 PM_Temperature_Sensor_Buffer_Enable

表 19-29 函数 PM_Temperature_Sensor_Buffer_Enable

函数名	PM_Temperature_Sensor_Buffer_Enable
函数原型	void PM_Temperature_Sensor_Buffer_Enable (FunctionalState NewState)
功能描述	设置 TEMPSENSOR 输出到 BUFFER 使能
输入参数 1	NewState: TEMPSENSOR 输出到 BUFFER 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.28 函数 PM_Reference_Voltage_Enable

表 19-30 函数 PM_Reference_Voltage_Enable

函数名	PM_Reference_Voltage_Enable
函数原型	void PM_Reference_Voltage_Enable (FunctionalState NewState)
功能描述	设置参考电压 2V 模块使能
输入参数 1	NewState: 参考电压 2V 模块使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.29 函数 PM_Internal_Test_Buffer_Clock_Enable

表 19-31 函数 PM_Internal_Test_Buffer_Clock_Enable

函数名	PM_Internal_Test_Buffer_Clock_Enable
函数原型	void PM_Internal_Test_Buffer_Clock_Enable (FunctionalState NewState)
功能描述	设置内部测试输出 BUFFER 时钟使能
输入参数 1	NewState: 内部测试输出 BUFFER 时钟使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.30 函数 PM_Internal_Test_Buffer_Clock_Scaler_Config

表 19-32 函数 PM_Internal_Test_Buffer_Clock_Scaler_Config

函数名	PM_Internal_Test_Buffer_Clock_Scaler_Config
函数原型	void PM_Internal_Test_Buffer_Clock_Scaler_Config (uint32_t SclkScaler)
功能描述	设置内部测试输出 BUFFER 工作时钟分频
输入参数 1	SclkScaler: 内部测试输出 BUFFER 工作时钟分频, 取值为: PM_BUFFER_SCLK_DIV_1: SCLK PM_BUFFER_SCLK_DIV_2: SCLK/2

	PM_BUFFER_SCLK_DIV_4: SCLK/4 PM_BUFFER_SCLK_DIV_8: SCLK/8 PM_BUFFER_SCLK_DIV_16: SCLK/16 PM_BUFFER_SCLK_DIV_32: SCLK/32 PM_BUFFER_SCLK_DIV_64: SCLK/64 PM_BUFFER_SCLK_DIV_128: SCLK/128
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.31 函数 PM_PLL0_Output_Buffer_Enable

表 19-33 函数 PM_PLL0_Output_Buffer_Enable

函数名	PM_PLL0_Output_Buffer_Enable
函数原型	void PM_PLL0_Output_Buffer_Enable (FunctionalState NewState)
功能描述	设置 PLL0 时钟输出到 BUFFER 使能
输入参数 1	NewState: PLL0 时钟输出到 BUFFER 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.32 函数 PM_PLL1_Output_Buffer_Enable

表 19-34 函数 PM_PLL1_Output_Buffer_Enable

函数名	PM_PLL1_Output_Buffer_Enable
函数原型	void PM_PLL1_Output_Buffer_Enable (FunctionalState NewState)
功能描述	设置 PLL1 时钟输出到 BUFFER 使能
输入参数 1	NewState: PLL1 时钟输出到 BUFFER 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.33 函数 PM_PLL2_Output_Buffer_Enable

表 19-35 函数 PM_PLL2_Output_Buffer_Enable

函数名	PM_PLL2_Output_Buffer_Enable
函数原型	void PM_PLL2_Output_Buffer_Enable (FunctionalState NewState)
功能描述	设置 PLL2 时钟输出到 BUFFER 使能
输入参数 1	NewState: PLL2 时钟输出到 BUFFER 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.34 函数 PM_PLL0LDO_Output_Buffer_Enable

表 19-36 函数 PM_PLL0LDO_Output_Buffer_Enable

函数名	PM_PLL0LDO_Output_Buffer_Enable
函数原型	void PM_PLL0LDO_Output_Buffer_Enable (FunctionalState NewState)
功能描述	设置 PLL0_LDO 输出到 BUFFER 使能
输入参数 1	NewState: PLL0_LDO 输出到 BUFFER 使能状态,取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.35 函数 PM_PLL1LDO_Output_Buffer_Enable

表 19-37 函数 PM_PLL1LDO_Output_Buffer_Enable

函数名	PM_PLL1LDO_Output_Buffer_Enable
函数原型	void PM_PLL1LDO_Output_Buffer_Enable (FunctionalState NewState)
功能描述	设置 PLL1_LDO 输出到 BUFFER 使能
输入参数 1	NewState: PLL1_LDO 输出到 BUFFER 使能状态,取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.36 函数 PM_PLL2LDO_Output_Buffer_Enable

表 19-38 函数 PM_PLL2LDO_Output_Buffer_Enable

函数名	PM_PLL2LDO_Output_Buffer_Enable
函数原型	void PM_PLL2LDO_Output_Buffer_Enable (FunctionalState NewState)
功能描述	设置 PLL2_LDO 输出到 BUFFER 使能
输入参数 1	NewState: PLL2_LDO 输出到 BUFFER 使能状态,取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.37 函数 PM_Battery_BOR_Config

表 19-39 函数 PM_Battery_BOR_Config

函数名	PM_Battery_BOR_Config
函数原型	void PM_Battery_BOR_Config (uint32_t Voltage)
功能描述	设置 BAT_BOR 电压点选择
输入参数 1	Voltage: BAT_BOR 电压点选择, 取值为: PM_BATTERY_VOLTAGE_1P6V: 1.6V PM_BATTERY_VOLTAGE_1P8V: 1.8V PM_BATTERY_VOLTAGE_2P06V: 2.06V

	PM_BATTERY_VOLTAGE_2P4V: 2.4V PM_BATTERY_VOLTAGE_2P88V: 2.88V PM_BATTERY_VOLTAGE_3P6V: 3.6V
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.38 函数 PM_Battery_BOR_Enable

表 19-40 函数 PM_Battery_BOR_Enable

函数名	PM_Battery_BOR_Enable
函数原型	void PM_Battery_BOR_Enable (FunctionalState NewState)
功能描述	设置 BAT_BOR 使能
输入参数 1	NewState: BAT_BOR 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.39 函数 PM_Peripheral_Voltage_Monitoring_Enable

表 19-41 函数 PM_Peripheral_Voltage_Monitoring_Enable

函数名	PM_Peripheral_Voltage_Monitoring_Enable
函数原型	void PM_Peripheral_Voltage_Monitoring_Enable (FunctionalState NewState)
功能描述	设置 PVM1 使能
输入参数 1	NewState: PVM1 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.40 函数 PM_Voltage_Detection_Config

表 19-42 函数 PM_Voltage_Detection_Config

函数名	PM_Voltage_Detection_Config
函数原型	void PM_Voltage_Detection_Config (uint32_t Voltage)
功能描述	设置电压检测点选择
输入参数 1	Voltage: 电压检测点选择, 取值为: PM_VOLTAGE_DETECTION_2P1V: 2.1V PM_VOLTAGE_DETECTION_2P25V: 2.25V PM_VOLTAGE_DETECTION_2P4V: 2.4V PM_VOLTAGE_DETECTION_2P55V: 2.55V PM_VOLTAGE_DETECTION_2P7V: 2.7V PM_VOLTAGE_DETECTION_2P85V: 2.85V

	PM_VOLTAGE_DETECTION_2P95V: 2.95V
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.41 函数 PM_Voltage_Detection_Enable

表 19-43 函数 PM_Voltage_Detection_Enable

函数名	PM_Voltage_Detection_Enable
函数原型	void PM_Voltage_Detection_Enable (FunctionalState NewState)
功能描述	设置电压检测功能使能
输入参数 1	NewState: 电压检测功能使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.42 函数 PM_External_Wakeup_Pin_Enable

表 19-44 函数 PM_External_Wakeup_Pin_Enable

函数名	PM_External_Wakeup_Pin_Enable
函数原型	void PM_External_Wakeup_Pin_Enable (uint32_t PinSel, FunctionalState NewState)
功能描述	设置外部唤醒引脚 WKPx 使能
输入参数 1	PinSel: 唤醒引脚, 取值为: PM_PIN_WKP1: 引脚 WKP1 PM_PIN_WKP2: 引脚 WKP2 PM_PIN_WKP3: 引脚 WKP3 PM_PIN_WKP4: 引脚 WKP4 PM_PIN_WKP5: 引脚 WKP5
输入参数 2	NewState: 外部唤醒引脚 WKPx 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.43 函数 PM_External_Wakeup_Edge_Config

表 19-45 函数 PM_External_Wakeup_Edge_Config

函数名	PM_External_Wakeup_Edge_Config
函数原型	void PM_External_Wakeup_Edge_Config (uint32_t PinSel, uint32_t TriggerEdge)
功能描述	设置外部唤醒引脚 WKPx 唤醒触发沿
输入参数 1	PinSel: 唤醒引脚, 取值为:

	PM_PIN_WKP1: 引脚 WKP1 PM_PIN_WKP2: 引脚 WKP2 PM_PIN_WKP3: 引脚 WKP3 PM_PIN_WKP4: 引脚 WKP4 PM_PIN_WKP5: 引脚 WKP5
输入参数 2	TriggerEdge: 外部唤醒引脚 WKPx 唤醒触发沿, 取值为: PM_TRIGGER_RISE_EDGE: WKPx 上升沿触发 PM_TRIGGER_FALL_EDGE: WKPx 下降沿触发
返回值	无
被调用函数	无

19.4.44 函数 PM_Stop_Mode_Peripheral_INLF_Enable

表 19-46 函数 PM_Stop_Mode_Peripheral_INLF_Enable

函数名	PM_Stop_Mode_Peripheral_INLF_Enable
函数原型	void PM_Stop_Mode_Peripheral_INLF_Enable (uint32_t Peripheral, FunctionalState NewState)
功能描述	设置不同外设工作在 Stop 模式下内部低频时钟使能
输入参数 1	Peripheral: 外设选择, 取值为: PM_PERIPHERAL_CAN0: CAN0 PM_PERIPHERAL_CCP: CCP PM_PERIPHERAL_LCD: LCD PM_PERIPHERAL_UART0: UART0
输入参数 2	NewState: 内部低频时钟使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.45 函数 PM_Peripheral_Reset_Config

表 19-47 函数 PM_Peripheral_Reset_Config

函数名	PM_Peripheral_Reset_Config
函数原型	void PM_Peripheral_Reset_Config (uint32_t Peripheral, uint32_t ResetStatus)
功能描述	设置不同外设复位
输入参数 1	Peripheral: 外设选择, 取值为: PM_PERIPHERAL_CAN0: CAN0 PM_PERIPHERAL_CCP: CCP PM_PERIPHERAL_LCD: LCD PM_PERIPHERAL_UART0: UART0
输入参数 2	ResetStatus: 外设复位状态, 取值为: PERIPHERAL_RST_STATUS: 外设处于复位状态 PERIPHERAL_OUTRST_STATUS: 外设退出复位

返回值	无
被调用函数	无

19.4.46 函数 PM_Vdd_Por_Enable

表 19-48 函数 PM_Vdd_Por_Enable

函数名	PM_Vdd_Por_Enable
函数原型	void PM_Vdd_Por_Enable (FunctionalState NewState)
功能描述	设置主电源 POR 强制使能
输入参数 1	NewState: 主电源 POR 强制使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.47 函数 PM_Low_Power_Bandgap_Enable

表 19-49 函数 PM_Low_Power_Bandgap_Enable

函数名	PM_Low_Power_Bandgap_Enable
函数原型	void PM_Low_Power_Bandgap_Enable (FunctionalState NewState)
功能描述	设置 LP_BG 软件使能
输入参数 1	NewState: LP_BG 软件使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.48 函数 PM_Power_Dissipation_Mode_Config

表 19-50 函数 PM_Power_Dissipation_Mode_Config

函数名	PM_Power_Dissipation_Mode_Config
函数原型	void PM_Power_Dissipation_Mode_Config (uint32_t Mode)
功能描述	设置功耗模式时序选择
输入参数 1	Mode: 功耗模式时序选择, 取值为: PM_POWER DISSIPATION_SCHEME1: Scheme1, 新增简易时序 PM_POWER DISSIPATION_SCHEME2: Scheme2, 时序参考 4003, 较复杂
返回值	无
被调用函数	无

19.4.49 函数 PM_Power_Dissipation_Mode_Delay_Config

表 19-51 函数 PM_Power_Dissipation_Mode_Delay_Config

函数名	PM_Power_Dissipation_Mode_Delay_Config
-----	--

函数原型	void PM_Power_Dissipation_Mode_Delay_Config (uint32_t DelayTime)
功能描述	设置 POR12 关闭到 LDO 关闭的延时时间
输入参数 1	DelayTime: POR12 关闭到 LDO 关闭的延时时间配置，取值为： PM_POWER DISSIPATION_500ns: 500ns PM_POWER DISSIPATION_500ns_1T: 500ns+1* T(INTLFOSC) PM_POWER DISSIPATION_500ns_2P5T: 500ns+2.5* T(INTLFOSC)
返回值	无
被调用函数	无

19.4.50 函数 PM_Internal_Test_Buffer_Enable

表 19-52 函数 PM_Internal_Test_Buffer_Enable

函数名	PM_Internal_Test_Buffer_Enable
函数原型	void PM_Internal_Test_Buffer_Enable (FunctionalState NewState)
功能描述	设置内部测试输出 BUFFER 使能
输入参数 1	NewState: 内部测试输出 BUFFER 使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.51 函数 PM_Clear_Reset_And_Wakeup_Flag

表 19-53 函数 PM_Clear_Reset_And_Wakeup_Flag

函数名	PM_Clear_Reset_And_Wakeup_Flag
函数原型	void PM_Clear_Reset_And_Wakeup_Flag (uint32_t EventSel)
功能描述	清零复位/唤醒状态标志位
输入参数 1	EventSel: 复位/唤醒事件，取值为： PM_RESET_IWDT: 独立看门狗复位事件 PM_WAKEUP_RTC_ALARM: RTC 闹钟唤醒事件 PM_WAKEUP_EXTERNAL_PIN: 外部唤醒引脚唤醒事件 PM_RESET_POR: POR 复位事件 PM_RESET_BOR: BOR 复位事件 PM_RESET_SOFTWARE: 软件复位事件
返回值	无
被调用函数	无

19.4.52 函数 PM_Get_IWDT_Reset_Flag

表 19-54 函数 PM_Get_IWDT_Reset_Flag

函数名	PM_Get_IWDT_Reset_Flag
函数原型	FlagStatus PM_Get_IWDT_Reset_Flag (uint32_t EventSel)

功能描述	获取对应复位/唤醒事件的状态标志
输入参数 1	EventSel: 复位/唤醒事件, 取值为: PM_RESET_IWDT: 独立看门狗复位事件 PM_WAKEUP_RTC_ALARM: RTC 闹钟唤醒事件 PM_WAKEUP_EXTERNAL_PIN: 外部唤醒引脚唤醒事件 PM_RESET_POR: POR 复位事件 PM_RESET_BOR: BOR 复位事件 PM_RESET_SOFTWARE: 软件复位事件 PM_WAKEUP_EXTERNAL_PIN_WKP5: WKP5 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP4: WKP4 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP3: WKP3 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP2: WKP2 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP1: WKP1 引脚唤醒事件
返回值	1: 发生了复位/唤醒事件; 0: 未发生复位/唤醒事件
被调用函数	无

19.4.53 函数 PM_Clear_External_Wakeup_Pin_Flag

表 19-55 函数 PM_Clear_External_Wakeup_Pin_Flag

函数名	PM_Clear_External_Wakeup_Pin_Flag
函数原型	void PM_Clear_External_Wakeup_Pin_Flag (uint32_t EventSel)
功能描述	清零外部唤醒引脚 WKP _x 状态标志
输入参数 1	EventSel: 复位/唤醒事件, 取值为: PM_WAKEUP_EXTERNAL_PIN_WKP5: WKP5 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP4: WKP4 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP3: WKP3 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP2: WKP2 引脚唤醒事件 PM_WAKEUP_EXTERNAL_PIN_WKP1: WKP1 引脚唤醒事件
返回值	无
被调用函数	无

19.4.54 函数 PM_Get_Low_Power_Running_State

表 19-56 函数 PM_Get_Low_Power_Running_State

函数名	PM_Get_Low_Power_Running_State
函数原型	FlagStatus PM_Get_Low_Power_Running_State (void)
功能描述	获取低功耗运行模式状态标志
输入参数 1	无
返回值	1: 电压调制器切换至 LPR; 0: 主电压调制器 MR 已使能完成
被调用函数	无

19.4.55 函数 PM_Get_LPR_Status

表 19-57 函数 PM_Get_LPR_Status

函数名	PM_Get_LPR_Status
函数原型	FlagStatus PM_Get_LPR_Status (void)
功能描述	获取低功耗电压调制器 LPR 状态
输入参数 1	无
返回值	1: 低功耗电压调制器 LPR 已使能完成; 0: 低功耗电压调制器 LPR 未使能完成
被调用函数	无

19.4.56 函数 PM_Get_Peripheral_Voltage_Detection_Status

表 19-58 函数 PM_Get_Peripheral_Voltage_Detection_Status

函数名	PM_Get_Peripheral_Voltage_Detection_Status
函数原型	FlagStatus PM_Get_Peripheral_Voltage_Detection_Status (void)
功能描述	获取电压检测状态
输入参数 1	无
返回值	1: VDD 电压高于 PVD 电压点; 0: VDD 电压低于 PVD 电压点
被调用函数	无

19.4.57 函数 PM_Zero_Drift_Current_Config

表 19-59 函数 PM_Zero_Drift_Current_Config

函数名	PM_Zero_Drift_Current_Config
函数原型	void PM_Zero_Drift_Current_Config (uint32_t Calibration)
功能描述	设置零温漂电流档校准
输入参数 1	Calibration: 零温漂电流档校准, 取值为: PM_CURRENT_OUTPUT_2uA: 输出电流 2uA PM_CURRENT_REDUCE_10_PERCENT: 电流减小加 10% PM_CURRENT_INCREASE_10_PERCENT: 电流增加 10% PM_CURRENT_INCREASE_20_PERCENT: 电流增加 20%
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.58 函数 PM_BOR_Voltage_Config

表 19-60 函数 PM_BOR_Voltage_Config

函数名	PM_BOR_Voltage_Config
函数原型	void PM_BOR_Voltage_Config (uint32_t Voltage)
功能描述	设置 BOR 电压点选择
输入参数 1	Voltage: BOR 电压点选择, 取值为: PM_BOR_VOLTAGE_2P05V: BOR 电压点为 2.05V PM_BOR_VOLTAGE_2P25V: BOR 电压点为 2.25V PM_BOR_VOLTAGE_2P55V: BOR 电压点为 2.55V PM_BOR_VOLTAGE_2P85V: BOR 电压点为 2.85V
返回值	无
被调用函数	无

19.4.59 函数 PM_Main_Regulator_Voltage_Config

表 19-61 函数 PM_Main_Regulator_Voltage_Config

函数名	PM_Main_Regulator_Voltage_Config
函数原型	void PM_Main_Regulator_Voltage_Config (uint32_t MRSel, uint32_t Voltage)
功能描述	设置 MR 或 MR_HV 模块校准
输入参数 1	MRSel: MR 或 MR_HV 模块选择, 取值为: PM_MR_MODULE: MR 模块 PM_MR_HV_MODULE: MR_HV 模块
输入参数 2	Voltage: 校准电压选择, 取值为: PM_MR_VOLTAGE_1P2V: BOR 电压点为 1.2V PM_MR_VOLTAGE_0P9V: BOR 电压点为 0.9V PM_MR_VOLTAGE_1V: BOR 电压点为 1V PM_MR_VOLTAGE_1P32V: BOR 电压点为 1.32V
返回值	无
被调用函数	无

19.4.60 函数 PM_Main_Regulator_HV_Enable

表 19-62 函数 PM_Main_Regulator_HV_Enable

函数名	PM_Main_Regulator_HV_Enable
函数原型	void PM_Main_Regulator_HV_Enable (FunctionalState NewState)
功能描述	设置 MR_HV 模块软件使能
输入参数 1	NewState: MR_HV 模块软件使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

19.4.61 函数 PM_Reference_Calibration_Config

表 19-63 函数 PM_Reference_Calibration_Config

函数名	PM_Reference_Calibration_Config
函数原型	void PM_Reference_Calibration_Config (uint32_t Reference, uint32_t Calibration)
功能描述	设置不同的基准校准
输入参数 1	Reference: 不同的基准选择, 取值为: PM_REFERENCE_BUFFER: 参考 BUFFER 基准校准 PM_REFERENCE_LDO12: LDO12 基准校准 PM_REFERENCE_LDO18: LDO18 基准校准
输入参数 2	Calibration: 校准电压选择, PM_REFERENCE_BUFFER 取值范围为 0~0x1F, PM_REFERENCE_LDO12 取值范围为 0~0xF, PM_REFERENCE_LDO18 取值范围为 0~0xF
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.62 函数 PM_INTLF_Bias_Current_Config

表 19-64 函数 PM_INTLF_Bias_Current_Config

函数名	PM_INTLF_Bias_Current_Config
函数原型	void PM_INTLF_Bias_Current_Config (uint32_t Calibration)
功能描述	设置内部低频振荡器电流校准
输入参数 1	Calibration: 偏置电流选择, 取值为: PM_INTLF_BIAS_CURRENT_10nA: 10nA PM_INTLF_BIAS_CURRENT_12P5nA: 12.5nA PM_INTLF_BIAS_CURRENT_15nA: 15nA PM_INTLF_BIAS_CURRENT_17P5nA: 17.5nA PM_INTLF_BIAS_CURRENT_0nA: 0nA PM_INTLF_BIAS_CURRENT_2P5nA: 2.5nA PM_INTLF_BIAS_CURRENT_5nA: 5nA PM_INTLF_BIAS_CURRENT_7P5nA: 7.5nA
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.63 函数 PM_EXTLF_Bias_Current_Config

表 19-65 函数 PM_EXTLF_Bias_Current_Config

函数名	PM_EXTLF_Bias_Current_Config
函数原型	void PM_EXTLF_Bias_Current_Config (uint32_t Calibration)
功能描述	设置外部低频振荡器电流校准
输入参数 1	Calibration: 偏置电流选择, 取值为: PM_EXTLF_BIAS_CURRENT_20nA: 20nA PM_EXTLF_BIAS_CURRENT_25nA: 25nA PM_EXTLF_BIAS_CURRENT_30nA: 30nA PM_EXTLF_BIAS_CURRENT_35nA: 35nA PM_EXTLF_BIAS_CURRENT_0nA: 0nA PM_EXTLF_BIAS_CURRENT_5nA: 5nA PM_EXTLF_BIAS_CURRENT_10nA: 10nA PM_EXTLF_BIAS_CURRENT_15nA: 15nA
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.64 函数 PM_INTLF_Capacitance_Calibration_Config

表 19-66 函数 PM_INTLF_Capacitance_Calibration_Config

函数名	PM_INTLF_Capacitance_Calibration_Config
函数原型	void PM_INTLF_Capacitance_Calibration_Config (uint32_t Calibration)
功能描述	设置内部低频振荡器电容校准
输入参数 1	Calibration: 校准值, 取值范围为 0~0x1F
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.65 函数 PM_LP_Bias_Calibration_Config

表 19-67 函数 PM_LP_Bias_Calibration_Config

函数名	PM_LP_Bias_Calibration_Config
函数原型	void PM_LP_Bias_Calibration_Config (uint32_t Calibration)
功能描述	设置低功耗偏置电流校准
输入参数 1	Calibration: 校准值, 取值范围为 0~0x7
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.66 函数 PM_LPBG_Pump_Calibration_Config

表 19-68 函数 PM_LPBG_Pump_Calibration_Config

函数名	PM_LPBG_Pump_Calibration_Config
函数原型	void PM_LPBG_Pump_Calibration_Config (uint32_t Calibration)
功能描述	设置内部低频振荡器电容校准
输入参数 1	Calibration: 校准值, 取值范围为 0~0x7
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.67 函数 PM_EXTLF_N_Bias_Current_Config

表 19-69 函数 PM_EXTLF_N_Bias_Current_Config

函数名	PM_EXTLF_N_Bias_Current_Config
函数原型	void PM_EXTLF_N_Bias_Current_Config (uint32_t Calibration)
功能描述	设置外部低频振荡器 N 管偏置校准
输入参数 1	Calibration: 电流选择, 取值为: PM_BRANCH_CURRENT_NONE: 没有增加支路电流 PM_BRANCH_CURRENT_50_PERCENT: 增加 50%支路电流 PM_BRANCH_CURRENT_150_PERCENT: 增加 150%支路电流 PM_BRANCH_CURRENT_200_PERCENT: 增加 200%支路电流
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

19.4.68 函数 PM_EXTLF_PIN_Selection_Config

表 19-70 函数 PM_EXTLF_PIN_Selection_Config

函数名	PM_EXTLF_PIN_Selection_Config
函数原型	void PM_EXTLF_PIN_Selection_Config (uint32_t PeripheralPort)
功能描述	设置外部低频振荡器引脚选择位
输入参数 1	PeripheralPort: 设置外部低频振荡器引脚 IO 口, 取值为: PM_EXTLF_PIN1_IO_PORT: 选择外部低频晶振 1 PM_EXTLF_PIN2_IO_PORT: 选择外部低频晶振 2
返回值	无
被调用函数	无

19.4.69 函数 PM_EXTHF_PIN_Selection_Config

表 19-71 函数 PM_EXTHF_PIN_Selection_Config

函数名	PM_EXTHF_PIN_Selection_Config
函数原型	void PM_EXTHF_PIN_Selection_Config (uint32_t PeripheralPort)
功能描述	设置外部高频振荡器引脚选择位
输入参数 1	PeripheralPort: 设置外部高频振荡器引脚 IO 口, 取值为: PM_EXHLF_PIN1_IO_PORT: 选择外部高频晶振 1 PM_EXHLF_PIN2_IO_PORT: 选择外部高频晶振 2
返回值	无
被调用函数	无

19.4.70 函数 PM_LDO18_Module_Config

表 19-72 函数 PM_LDO18_Module_Config

函数名	PM_LDO18_Module_Config
函数原型	void PM_LDO18_Module_Config (uint32_t LDO18Config)
功能描述	设置 LDO18 内置模块选择
输入参数 1	LDO18Config: LDO18 内置模块选择, 取值为: PM_LDO18_CAP: LDO18 选择 cap 结构 PM_LDO18_CAPLESS: LDO18 选择 capless 结构
返回值	无
被调用函数	无

19.4.71 函数 PM_Main_Regulator_Bandgap_Config

表 19-73 函数 PM_Main_Regulator_Bandgap_Config

函数名	PM_Main_Regulator_Bandgap_Config
函数原型	void PM_Main_Regulator_Bandgap_Config (uint32_t ModeSel)
功能描述	设置 MR_BG 模式选择
输入参数 1	ModeSel: MR_BG 模式选择, 取值为: PM_LOW_POWER_MODE: 低功耗模式 PM_HIGH_POWER_MODE: 高功耗模式
返回值	无
被调用函数	无

19.4.72 函数 PM_LPR_Module_Config

表 19-74 函数 PM_LPR_Module_Config

函数名	PM_LPR_Module_Config
-----	----------------------

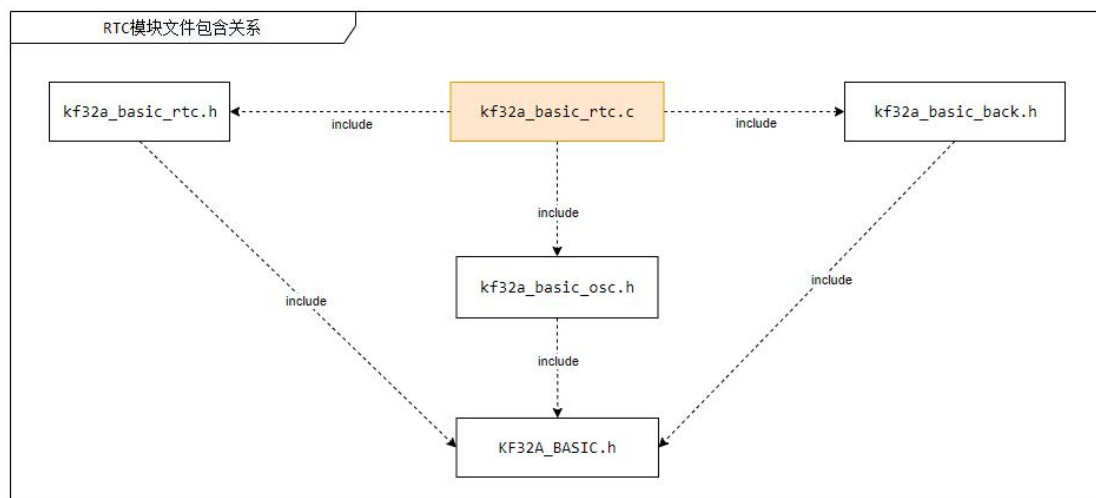
函数原型	void PM_LPR_Module_Config (uint32_t ModeSel)
功能描述	设置 LPR 内置模块选择
输入参数 1	ModeSel: LPR 内置模块选择，取值为: PM_LPR_DEFAULT: 低功耗模式 PM_LPR_BACKUP: 高功耗模式
返回值	无
被调用函数	无

20 实时时钟（RTC）

实时时钟(Real Time Counting, RTC)单元提供给用户实时时间以及日历信息。并可以根据年、月份（闰年、大小月），自动补偿天数；还可以进行夏令时、冬令时补偿。并提供了两个可编程闹钟。

20.1 文件引用关系

图 20-1 RTC 模块文件包含关系



20.2 RTC 寄存器结构

RTC 寄存器结构，RTC_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct RTC_MemMap {
    volatile uint32_t    CR;
    volatile uint32_t    ALRA;
    volatile uint32_t    TMR;
    volatile uint32_t    DTR;
    volatile uint32_t    ALRB;
    volatile uint32_t    TMER;
    volatile uint32_t    TCR;
    volatile uint32_t    IER;
    volatile uint32_t    IFR;
    volatile uint32_t    TMBR;
    volatile uint32_t    DTBR;
}RTC_SFRmap;
    
```

表 20-1 SYSTICK 寄存器结构说明

寄存器	描述
-----	----

CR	RTC 控制寄存器, 偏移:0x00
ALRA	RTC 闹钟 A 控制寄存器, 偏移:0x4
TMR	RTC 时间寄存器, 偏移:0x8
DTR	RTC 日期寄存器, 偏移:0xC
ALRB	RTC 闹钟 B 控制寄存器, 偏移:0x10
TMER	RTC 定时器寄存器, 偏移:0x14
TCR	RTC 定时器控制寄存器, 偏移:0x18
IER	RTC 中断使能寄存器, 偏移:0x1C
IFR	RTC 中断标志寄存器, 偏移:0x20
TMBR	RTC 时间备份寄存器, 偏移:0x24
DTBR	RTC 日期备份寄存器, 偏移:0x28

RTC 配置信息结构体, RTC_InitTypeDef, 定义于文件 kf32a_basic_rtc.h 中, 如下:

```
typedef struct
```

```
{
    uint32_t    m_ClockSource;
    uint32_t    m_HourFormat;
    RTC_TimeTypeDef    m_TimeStruct;
    RTC_DateTypeDef m    _DateStruct;
}RTC_InitTypeDef;
```

表 20-2 RTC 配置信息结构体说明

寄存器	描述
m_ClockSource	RTC时钟源,取值为宏“RTC时钟源”中的一个
m_HourFormat	时间格式,取值为宏“时间格式”中的一个。
m_TimeStruct	时间信息结构体
m_DateStruct	日期信息结构体

RTC 时间信息结构体, RTC_TimeTypeDef,定义于文件 kf32a_basic_rtc.h 中, 如下:

```
typedef struct
```

```
{
    uint8_t    m_Hours;
    uint8_t    m_Minutes;
    uint8_t    m_Seconds;
    uint8_t    m_AMPM;
}RTC_TimeTypeDef;
```

表 20-3 RTC 配置信息结构体说明

寄存器	描述
m_Hours	RTC 时钟, 取值范围 24 小时制时为 0-23, 12 小时制时为 1-12
m_Minutes	RTC 分钟, 取值范围为 0-59
m_Seconds	RTC 秒钟,值范围为 0-59
m_AMPM	RTC AM/PM 选择, 取值为宏“AM/PM 选择”中的一个

RTC 时间信息结构体，RTC_DataTypeDef,定义于文件 kf32a_basic_rtc.h 中，如下：

```
typedef struct
{
    uint8_t    m_WeekDay;
    uint8_t    m_Day;
    uint8_t    m_Month;
    uint8_t    m_Year;
}RTC_DataTypeDef;
```

表 20-4 RTC 配置信息结构体说明

寄存器	描述
m_WeekDay	RTC 星期，取值为宏“RTC 星期”中的一个。
m_Day	RTC 日，取值范围为 1-31
m_Month	RTC 月，取值为宏“RTC 月份”中的一个
m_Year	RTC 年，取值范围为 0-99

20.3 RTC 宏定义

RTC 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，RTC 其他相关宏定义详见文件 kf32a_basic_rtc.h 及函数参数描述。

20.4 RTC 库函数

表 20-5 RTC 固件库函数列表

序号	函数名	描述
1	RTC_Reset	复位 RTC 模块
2	RTC_Configuration	日期时间初始化配置，该函数结束时退出配置模式
3	RTC_Time_Struct_Init	初始化时间信息结构体
4	RTC_Date_Struct_Init	初始化日期信息结构体
5	RTC_Struct_Init	初始化配置信息结构体
6	RTC_Get_Time_Configuration	获取时间信息
7	RTC_Get_Date_Configuration	获取日期信息
8	RTC_Alarm_Configuration	闹钟初始化配置，配置此函数前需配置 HT 位，通过 RTC_Hour_Format_Config 函数实现
9	RTC_Alarm_Struct_Init	初始化闹钟信息结构体
10	RTC_Clock_Calibration_Config	配置 RTC 时钟校正值
11	RTC_Time_Tick_Output_Enable	设置 RTC 的周期时间节拍

		(TT) 输出使能
12	RTC_Time_Stamp_Edge_Config	设置时间戳触发沿
13	RTC_Time_Stamp_Edge_Enable	设置时间戳触发沿使能
14	RTC_Add_One_Hour_Enable	设置 RTC 时间增加 1 小时使能
15	RTC_Sub_One_Hour_Enable	设置 RTC 时间减少 1 小时使能
16	RTC_Time_Tick_Config	配置时间节拍
17	RTC_Start_Config	启动实时时钟
18	RTC_Reset_Config	初始化实时时钟模块
19	RTC_Get_Leap_Year_Flag	获取闰年指示标志
20	RTC_Hour_Format_Config	设置小时显示的类型
21	RTC_Config_Mode_Enable	设置配置模式使能
22	RTC_Get_Operation_Off_Flag	获取 RTC 操作关闭指示标志
23	RTC_Get_Action_State_Flag	获取 RTC 活动状态
24	RTC_Enable	设置 RTC 使能
25	RTC_Alarm_A_Enable	设置闹钟 A 使能
26	RTC_Alarm_A_Weekday_Enable	设置闹钟 A 周使能
27	RTC_Alarm_A_Weekday_Config	配置闹钟中断星期时间
28	RTC_Alarm_A_Hours_Enable	设置闹钟 A 时钟使能
29	RTC_Alarm_A_AMPM_Config	设置闹钟 A 上午下午选择
30	RTC_Alarm_A_Hours_Config	配置闹钟中断小时时间，用户控制匹配 12 小时制或 24 小时制
31	RTC_Alarm_A_Minutes_Enable	设置闹钟 A 分钟使能
32	RTC_Alarm_A_Minutes_Config	配置闹钟中断分钟时间
33	RTC_Alarm_A_Seconds_Enable	设置闹钟 A 秒使能
34	RTC_Alarm_A_Seconds_Config	配置闹钟中断秒时间
35	RTC_Alarm_B_Enable	设置闹钟 B 使能
36	RTC_Alarm_B_Weekday_Enable	设置闹钟 B 周使能
37	RTC_Alarm_B_Weekday_Config	配置闹钟中断星期时间
38	RTC_Alarm_B_Hours_Enable	设置闹钟 B 时钟使能
39	RTC_Alarm_B_AMPM_Config	设置闹钟 B 上午下午选择
40	RTC_Alarm_B_Hours_Config	配置闹钟中断小时时间，用户控制匹配 12 小时制或 24 小时制
41	RTC_Alarm_B_Minutes_Enable	设置闹钟 B 分钟使能
42	RTC_Alarm_B_Minutes_Config	配置闹钟中断分钟时间
43	RTC_Alarm_B_Seconds_Enable	设置闹钟 B 秒使能
44	RTC_Alarm_B_Seconds_Config	配置闹钟中断秒时间
45	RTC_Weekday_Config	配置 RTC 时钟星期
46	RTC_AMPM_Config	设置 RTC 时钟上午下午选择
47	RTC_Hours_Config	配置 RTC 时钟时钟时间

48	RTC_Minutes_Config	配置 RTC 时钟分钟时间
49	RTC_Seconds_Config	配置 RTC 时钟秒钟时间
50	RTC_Year_Config	配置 RTC 时钟年份
51	RTC_Month_Config	配置 RTC 时钟月份
52	RTC_Day_Config	配置 RTC 时钟日期
53	RTC_Weekday_Backup_Config	配置 RTC 时钟星期备份
54	RTC_AMPM_Backup_Config	设置 RTC 时钟上午下午选择备份
55	RTC_Hours_Backup_Config	配置 RTC 时钟时钟时间备份
56	RTC_Minutes_Backup_Config	配置 RTC 时钟分钟时间备份
57	RTC_Seconds_Backup_Config	配置 RTC 时钟秒钟时间备份
58	RTC_Year_Backup_Config	配置 RTC 时钟年份备份
59	RTC_Month_Backup_Config	配置 RTC 时钟月份备份
60	RTC_Day_Backup_Config	配置 RTC 时钟日期备份
61	RTC_Timer1_Config	配置 RTC 定时器 1 计数值
62	RTC_Timer0_Config	配置 RTC 定时器 0 计数值
63	RTC_Timer1_Enable	设置 RTC 定时器 1 使能
64	RTC_Timer0_Enable	设置 RTC 定时器 0 使能
65	RTC_Timer1_Source_Config	配置定时器 1 时钟源选择
66	RTC_Timer0_Source_Config	配置定时器 0 时钟源选择
67	RTC_Time_Stamp_INT_Enable	设置 RTC 时间戳中断使能
68	RTC_Time_Stamp_Overflow_INT_Enable	设置 RTC 时间戳溢出中断使能
69	RTC_Timer1_INT_Enable	设置 RTC 定时器 1 中断使能
70	RTC_Timer0_INT_Enable	设置 RTC 定时器 0 中断使能
71	RTC_Time_Tick_INT_Enable	设置 RTC 时间节拍中断使能
72	RTC_Alarm_B_INT_Enable	设置 RTC 闹钟 B 中断使能
73	RTC_Alarm_A_INT_Enable	设置 RTC 闹钟 A 中断使能
74	RTC_Days_INT_Enable	设置 RTC 日进程中中断使能
75	RTC_Hours_INT_Enable	设置 RTC 小时进程中中断使能
76	RTC_Minutes_INT_Enable	设置 RTC 分进程中中断使能
77	RTC_Seconds_INT_Enable	设置 RTC 秒进程中中断使能
78	RTC_Get_Time_Stamp_INT_Flag	获取时间戳中断标志
79	RTC_Get_Time_Stamp_Overflow_INT_Flag	获取时间戳溢出中断标志
80	RTC_Get_Timer1_INT_Flag	获取 RTC 定时器 1 中断标志
81	RTC_Get_Timer0_INT_Flag	获取 RTC 定时器 0 中断标志
82	RTC_Get_Time_Tick_INT_Flag	获取 RTC 时间节拍中断标志
83	RTC_Get_Alarm_B_INT_Flag	获取 RTC 闹钟 B 中断标志
84	RTC_Get_Alarm_A_INT_Flag	获取 RTC 闹钟 A 中断标志
85	RTC_Get_Days_INT_Flag	获取 RTC 日进程中中断标志
86	RTC_Get_Hours_INT_Flag	获取 RTC 小时进程中中断标志
87	RTC_Get_Minutes_INT_Flag	获取 RTC 分进程中中断标志

88	RTC_Get_Seconds_INT_Flag	获取 RTC 秒进程中断标志
89	RTC_Clear_Time_Stamp_INT_Flag	清零 RTC 时间戳中断标志
90	RTC_Clear_Time_Stamp_Overflow_INT_Flag	清零 RTC 时间戳溢出中断标志
91	RTC_Clear_Timer1_INT_Flag	清零 RTC 定时器 1 中断标志
92	RTC_Clear_Timer0_INT_Flag	清零 RTC 定时器 0 中断标志
93	RTC_Clear_Time_Tick_INT_Flag	清零时间节拍中断标志
94	RTC_Clear_Alarm_B_INT_Flag	清零闹钟 B 中断标志
95	RTC_Clear_Alarm_A_INT_Flag	清零闹钟 A 中断标志
96	RTC_Clear_Days_INT_Flag	清零 RTC 日进程中断标志
97	RTC_Clear_Hours_INT_Flag	清零 RTC 小时进程中断标志
98	RTC_Clear_Minutes_INT_Flag	清零 RTC 分进程中断标志
99	RTC_Clear_Seconds_INT_Flag	清零 RTC 秒进程中断标志

20.4.1 函数 RTC_Reset

表 20-6 函数 RTC_Reset

函数名	RTC_Reset
函数原型	void RTC_Reset (void)
功能描述	复位 RTC 模块
输入参数 1	无
返回值	无
被调用函数	无

20.4.2 函数 RTC_Configuration

表 20-7 函数 RTC_Configuration

函数名	RTC_Configuration
函数原型	void RTC_Configuration (uint32_t TimeFormat, RTC_InitTypeDef * rtcInitStruct)
功能描述	日期时间初始化配置，该函数结束时会退出配置模式
输入参数 1	TimeFormat: 日期时间数值格式，选择是否为 BCD 码，取值范围为： RTC_TIME_FORMAT_BCD: BCD 编码方式 RTC_TIME_FORMAT_BIN: 不使用 BCD 编码
输入参数 2	rtcInitStruct: 初始化信息结构体指针
返回值	无
被调用函数	无

20.4.3 函数 RTC_Time_Struct_Init

表 20-8 函数 RTC_Time_Struct_Init

函数名	RTC_Time_Struct_Init
函数原型	void RTC_Time_Struct_Init (RTC_TimeTypeDef* rtcTimeInitStruct)
功能描述	初始化时间信息结构体
输入参数 1	rtcTimeInitStruct:指向待初始化的结构体指针
返回值	无
被调用函数	无

20.4.4 函数 RTC_Date_Struct_Init

表 20-9 函数 RTC_Date_Struct_Init

函数名	RTC_Date_Struct_Init
函数原型	void RTC_Date_Struct_Init (RTC_DateTypeDef* rtcDateInitStruct)
功能描述	初始化日期信息结构体
输入参数 1	rtcDateInitStruct:指向待初始化的结构体指针
返回值	无
被调用函数	无

20.4.5 函数 RTC_Struct_Init

表 20-10 函数 RTC_Struct_Init

函数名	RTC_Struct_Init
函数原型	void RTC_Struct_Init (RTC_InitTypeDef * rtcInitStruct)
功能描述	初始化配置信息结构体
输入参数 1	rtcInitStruct:指向待初始化的结构体指针
返回值	无
被调用函数	无

20.4.6 函数 RTC_Get_Time_Configuration

表 20-11 函数 RTC_Get_Time_Configuration

函数名	RTC_Get_Time_Configuration
函数原型	Void RTC_Get_Time_Configuration (uint32_t TimeFormat, RTC_TimeTypeDef* rtcTimeStruct)
功能描述	获取时间信息
输入参数 1	TimeFormat: 日期时间数值格式，选择是否为 BCD 码，取值范围为： RTC_TIME_FORMAT_BCD:BCD 编码方式 RTC_TIME_FORMAT_BIN:不使用 BCD 编码

输入参数 2	rtcTimeStruct:时间信息存储指针
返回值	无
被调用函数	无

20.4.7 函数 RTC_Get_Date_Configuration

表 20-12 函数 RTC_Get_Date_Configuration RTC_Alarm_Configuration

函数名	RTC_Get_Date_Configuration
函数原型	void RTC_Get_Date_Configuration (uint32_t TimeFormat, RTC_DateTypeDef* rtcDateStruct)
功能描述	获取日期信息
输入参数 1	TimeFormat: 日期时间数值格式，选择是否为 BCD 码，取值范围为： RTC_TIME_FORMAT_BCD:BCD 编码方式 RTC_TIME_FORMAT_BIN:不使用 BCD 编码
输入参数 1	rtcDateStruct:日期信息存储指针
返回值	无
被调用函数	无

20.4.8 函数 RTC_Alarm_Configuration

表 20-13 函数 RTC_Alarm_Configuration RTC_Alarm_Struct_Init

函数名	RTC_Alarm_Configuration
函数原型	void RTC_Alarm_Configuration (uint32_t AlarmSelect, uint32_t TimeFormat, RTC_AlarmTypeDef* rtcAlarmInitStruct)
功能描述	闹钟初始化配置，配置此函数前需配置 HT 位，通过 RTC_Hour_Format_Config 函数实现
输入参数 1	AlarmSelect:闹钟选择，取值范围为： RTC_ALARM_A_ADDR_OFFSET:闹钟 A， RTC_ALARM_B_ADDR_OFFSET:闹钟 B
输入参数 2	TimeFormat:日期时间数值格式，选择是否为 BCD 码，取值范围为： RTC_TIME_FORMAT_BCD:BCD 编码方式 RTC_TIME_FORMAT_BIN:不使用 BCD 编码
输入参数 3	rtcAlarmInitStruct:闹钟信息结构体指针
返回值	无
被调用函数	无

20.4.9 函数 RTC_Alarm_Struct_Init

表 20-14 函数 RTC_Alarm_Struct_Init RTC_Clock_Calibration_Config

函数名	RTC_Alarm_Struct_Init
-----	-----------------------

函数原型	void RTC_Alarm_Struct_Init (RTC_AlarmTypeDef* rtcAlarmInitStruct)
功能描述	初始化闹钟信息结构体
输入参数 1	rtcAlarmInitStruct:指向待初始化的结构体指针
返回值	无
被调用函数	无

20.4.10 函数 RTC_Clock_Calibration_Config

表 20-15 函数 RTC_Clock_Calibration_Config

函数名	RTC_Clock_Calibration_Config
函数原型	void RTC_Clock_Calibration_Config (int8_t Calibration)
功能描述	配置 RTC 时钟校正
输入参数 1	Calibration:RTC 时钟偏差的值, 取值为 8 位有符号数值
返回值	无
被调用函数	无

20.4.11 函数 RTC_Time_Tick_Output_Enable

表 20-16 函数 RTC_Time_Tick_Output_Enable

函数名	RTC_Time_Tick_Output_Enable
函数原型	void RTC_Time_Tick_Output_Enable (FunctionalState NewState)
功能描述	设置 RTC 的周期时间节拍 (TT) 输出使能
输入参数 1	NewState:RTC 的周期时间节拍 (TT) 输出使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.12 函数 RTC_Time_Stamp_Edge_Config

表 20-17 函数 RTC_Time_Stamp_Edge_Config

函数名	RTC_Time_Stamp_Edge_Config
函数原型	void RTC_Time_Stamp_Edge_Config (uint32_t TimeStamp)
功能描述	设置时间戳触发沿
输入参数 1	TimeStamp:时间戳触发沿, 取值为: RTC_TIME_STAMP_RISE:上升沿触发 RTC_TIME_STAMP_FALL:下降沿触发
返回值	无
被调用函数	无

20.4.13 函数 RTC_Time_Stamp_Edge_Enable

表 20-18 函数 RTC_Time_Stamp_Edge_Enable

函数名	RTC_Time_Stamp_Edge_Enable
函数原型	void RTC_Time_Stamp_Edge_Enable (FunctionalState NewState)
功能描述	设置时间戳触发沿使能
输入参数 1	NewState:时间戳触发沿使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.14 函数 RTC_Add_One_Hour_Enable

表 20-19 函数 RTC_Add_One_Hour_Enable

函数名	RTC_Add_One_Hour_Enable
函数原型	void RTC_Add_One_Hour_Enable (FunctionalState NewState)
功能描述	设置 RTC 时间增加 1 小时使能
输入参数 1	NewState:RTC 时间增加 1 小时使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.15 函数 RTC_Sub_One_Hour_Enable

表 20-20 函数 RTC_Sub_One_Hour_Enable

函数名	RTC_Sub_One_Hour_Enable
函数原型	void RTC_Sub_One_Hour_Enable (FunctionalState NewState)
功能描述	设置 RTC 时间减少 1 小时使能
输入参数 1	NewState:RTC 时间减少 1 小时使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.16 函数 RTC_Time_Tick_Config

表 20-21 函数 RTC_Time_Tick_Config

函数名	RTC_Time_Tick_Config
函数原型	void RTC_Time_Tick_Config (uint32_t Calibration)
功能描述	配置时间节拍
输入参数 1	TimeTick:RTC 时间节拍，取值为: RTC_TIME_TICK_DIV_1:时间节拍为 1 秒 RTC_TIME_TICK_DIV_2:时间节拍为 12 秒 RTC_TIME_TICK_DIV_4:时间节拍为 14 秒

	RTC_TIME_TICK_DIV_8:时间节拍为 18 秒 RTC_TIME_TICK_DIV_16:时间节拍为 116 秒 RTC_TIME_TICK_DIV_32:时间节拍为 132 秒 RTC_TIME_TICK_DIV_64:时间节拍为 164 秒 RTC_TIME_TICK_DIV_128:时间节拍为 1128 秒
返回值	无
被调用函数	无

20.4.17 函数 RTC_Start_Config

表 20-22 函数 RTC_Start_Config

函数名	RTC_Start_Config
函数原型	void RTC_Start_Config (void)
功能描述	启动实时时钟
输入参数 1	无
返回值	无
被调用函数	无

20.4.18 函数 RTC_Reset_Config

表 20-23 函数 RTC_Reset_Config

函数名	RTC_Reset_Config
函数原型	void RTC_Reset_Config (void)
功能描述	初始化实时时钟模块
输入参数 1	无
返回值	无
被调用函数	无

20.4.19 函数 RTC_Get_Leap_Year_Flag

表 20-24 函数 RTC_Get_Leap_Year_Flag

函数名	RTC_Get_Leap_Year_Flag
函数原型	FlagStatus RTC_Get_Leap_Year_Flag (void)
功能描述	获取闰年指示标志
输入参数 1	无
返回值	1:当前年份为闰年；0:当前年份为平年
被调用函数	无

20.4.20 函数 RTC_Hour_Format_Config

表 20-25 函数 RTC_Hour_Format_Config

函数名	RTC_Hour_Format_Config
函数原型	void RTC_Hour_Format_Config (uint32_t HourFormat)
功能描述	设置小时显示的类型
输入参数 1	HourFormat:RTC 小时显示的类型，取值为: RTC_HOUR_FORMAT_24:24 小时制 RTC_HOUR_FORMAT_12:12 小时制
返回值	无
被调用函数	无

20.4.21 函数 RTC_Config_Mode_Enable

表 20-26 函数 RTC_Config_Mode_Enable

函数名	RTC_Config_Mode_Enable
函数原型	void RTC_Config_Mode_Enable (FunctionalState ConfigMode)
功能描述	设置配置模式使能
输入参数 1	ConfigMode:配置模式状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.22 函数 RTC_Get_Operation_Off_Flag

表 20-27 函数 RTC_Get_Operation_Off_Flag

函数名	RTC_Get_Operation_Off_Flag
函数原型	FlagStatus RTC_Get_Operation_Off_Flag (void)
功能描述	获取 RTC 操作关闭指示标志
输入参数 1	无
返回值	1:上一次对 RTC 寄存器的写操作已经完成 0:上一次对 RTC 寄存器的写操作仍在进行
被调用函数	无

20.4.23 函数 RTC_Get_Action_State_Flag

表 20-28 函数 RTC_Get_Action_State_Flag

函数名	RTC_Get_Action_State_Flag
函数原型	FlagStatus RTC_Get_Action_State_Flag (void)
功能描述	获取 RTC 活动状态
输入参数 1	无

返回值	1:RTC 处在正常有效状态 0:RTC 处在复位状态
被调用函数	无

20.4.24 函数 RTC_Enable

表 20-29 函数 RTC_Enable

函数名	RTC_Enable
函数原型	void RTC_Enable (FunctionalState NewState)
功能描述	设置 RTC 使能
输入参数 1	NewState:RTC 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.25 函数 RTC_Alarm_A_Enable

表 20-30 函数 RTC_Alarm_A_Enable

函数名	RTC_Alarm_A_Enable
函数原型	void RTC_Alarm_A_Enable (FunctionalState NewState)
功能描述	设置闹钟 A 使能
输入参数 1	NewState:闹钟 A 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.26 函数 RTC_Alarm_A_Weekday_Enable

表 20-31 函数 RTC_Alarm_A_Weekday_Enable

函数名	RTC_Alarm_A_Weekday_Enable
函数原型	void RTC_Alarm_A_Weekday_Enable (FunctionalState NewState)
功能描述	设置闹钟 A 周使能
输入参数 1	NewState:闹钟 A 周使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.27 函数 RTC_Alarm_A_Weekday_Config

表 20-32 函数 RTC_Alarm_A_Weekday_Config

函数名	RTC_Alarm_A_Weekday_Config
函数原型	void RTC_Alarm_A_Weekday_Config (uint8_t Weekday)

功能描述	配置闹钟中断星期时间
输入参数 1	Weekday:闹钟中断星期时间, 取值为: RTC_WEEKDAY_MONDAY:星期一 RTC_WEEKDAY_TUESDAY:星期二 RTC_WEEKDAY_WEDNESDAY:星期三 RTC_WEEKDAY_THURSDAY:星期四 RTC_WEEKDAY_FRIDAY:星期五 RTC_WEEKDAY_SATURDAY:星期六 RTC_WEEKDAY_SUNDAY:星期天
返回值	无
被调用函数	无

20.4.28 函数 RTC_Alarm_A_Hours_Enable

表 20-33 函数 RTC_Alarm_A_Hours_Enable

函数名	RTC_Alarm_A_Hours_Enable
函数原型	void RTC_Alarm_A_Hours_Enable (FunctionalState NewState)
功能描述	设置闹钟 A 时钟使能
输入参数 1	NewState:闹钟 A 时钟使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.29 函数 RTC_Alarm_A_AMPM_Config

表 20-34 函数 RTC_Alarm_A_AMPM_Config

函数名	RTC_Alarm_A_AMPM_Config
函数原型	void RTC_Alarm_A_AMPM_Config (uint32_t NewSelect)
功能描述	设置闹钟 A 上午下午选择
输入参数 1	NewSelect:闹钟 A 上午下午选择, 取值为: RTC_TIME_AM:选择上午时间 RTC_TIME_PM:选择下午时间
返回值	无
被调用函数	无

20.4.30 函数 RTC_Alarm_A_Hours_Config

表 20-35 函数 RTC_Alarm_A_Hours_Config

函数名	RTC_Alarm_A_Hours_Config
函数原型	void RTC_Alarm_A_Hours_Config (uint32_t Hour)
功能描述	配置闹钟中断小时时间, 用户控制匹配 12 小时制或 24 小时制

输入参数 1	Hour:闹钟中断小时时间，取值匹配 12 小时制或 24 小时制
返回值	无
被调用函数	无

20.4.31 函数 RTC_Alarm_A_Minutes_Enable

表 20-36 函数 RTC_Alarm_A_Minutes_Enable

函数名	RTC_Alarm_A_Minutes_Enable
函数原型	void RTC_Alarm_A_Minutes_Enable (FunctionalState NewState)
功能描述	设置闹钟 A 分钟使能
输入参数 1	NewState:闹钟 A 分钟使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.32 函数 RTC_Alarm_A_Minutes_Config

表 20-37 函数 RTC_Alarm_A_Minutes_Config

函数名	RTC_Alarm_A_Minutes_Config
函数原型	void RTC_Alarm_A_Minutes_Config (uint32_t Minutes)
功能描述	配置闹钟中断分钟时间
输入参数 1	Minutes:闹钟中断分钟时间，取值为 0-59
返回值	无
被调用函数	无

20.4.33 函数 RTC_Alarm_A_Seconds_Enable

表 20-38 函数 RTC_Alarm_A_Seconds_Enable

函数名	RTC_Alarm_A_Seconds_Enable
函数原型	void RTC_Alarm_A_Seconds_Enable (FunctionalState NewState)
功能描述	设置闹钟 A 秒使能
输入参数 1	NewState:闹钟 A 秒使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.34 函数 RTC_Alarm_A_Seconds_Config

表 20-39 函数 RTC_Alarm_A_Seconds_Config

函数名	RTC_Alarm_A_Seconds_Config
函数原型	void RTC_Alarm_A_Seconds_Config (uint32_t Seconds)

功能描述	配置闹钟中断秒时间
输入参数 1	Seconds:闹钟中断秒时间，取值为 0-59
返回值	无
被调用函数	无

20.4.35 函数 RTC_Alarm_B_Enable

表 20-40 函数 RTC_Alarm_B_Enable

函数名	RTC_Alarm_B_Enable
函数原型	void RTC_Alarm_B_Enable (FunctionalState NewState)
功能描述	设置闹钟 B 使能
输入参数 1	NewState:闹钟 B 使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.36 函数 RTC_Alarm_B_Weekday_Enable

表 20-41 函数 RTC_Alarm_B_Weekday_Enable

函数名	RTC_Alarm_B_Weekday_Enable
函数原型	void RTC_Alarm_B_Weekday_Enable (FunctionalState NewState)
功能描述	设置闹钟 B 周使能
输入参数 1	NewState:闹钟 B 周使能状态，取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.37 函数 RTC_Alarm_B_Weekday_Config

表 20-42 函数 RTC_Alarm_B_Weekday_Config

函数名	RTC_Alarm_B_Weekday_Config
函数原型	void RTC_Alarm_B_Weekday_Config (uint8_t Weekday)
功能描述	配置闹钟中断星期时间
输入参数 1	Weekday:闹钟中断星期时间，取值为: RTC_WEEKDAY_MONDAY:星期一 RTC_WEEKDAY_TUESDAY:星期二 RTC_WEEKDAY_WEDNESDAY:星期三 RTC_WEEKDAY_THURSDAY:星期四 RTC_WEEKDAY_FRIDAY:星期五 RTC_WEEKDAY_SATURDAY:星期六 RTC_WEEKDAY_SUNDAY:星期天
返回值	无

被调用函数	无
-------	---

20.4.38 函数 RTC_Alarm_B_Hours_Enable

表 20-43 函数 RTC_Alarm_B_Hours_Enable

函数名	RTC_Alarm_B_Hours_Enable
函数原型	void RTC_Alarm_B_Hours_Enable (FunctionalState NewState)
功能描述	设置闹钟 B 时钟使能
输入参数 1	NewState:闹钟 B 时钟使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.39 函数 RTC_Alarm_B_AMPM_Config

表 20-44 函数 RTC_Alarm_B_AMPM_Config

函数名	RTC_Alarm_B_AMPM_Config
函数原型	void RTC_Alarm_B_AMPM_Config (uint32_t NewSelect)
功能描述	设置闹钟 B 上午下午选择
输入参数 1	NewSelect:闹钟 B 上午下午选择, 取值为: RTC_TIME_AM:选择上午时间 RTC_TIME_PM:选择下午时间
返回值	无
被调用函数	无

20.4.40 函数 RTC_Alarm_B_Hours_Config

表 20-45 函数 RTC_Alarm_B_Hours_Config

函数名	RTC_Alarm_B_Hours_Config
函数原型	void RTC_Alarm_B_Hours_Config (uint32_t Hour)
功能描述	配置闹钟中断小时时间, 用户控制匹配 12 小时制或 24 小时制
输入参数 1	Hour:闹钟中断小时时间, 取值匹配 12 小时制或 24 小时制
返回值	无
被调用函数	无

20.4.41 函数 RTC_Alarm_B_Minutes_Enable

表 20-46 函数 RTC_Alarm_B_Minutes_Enable

函数名	RTC_Alarm_B_Minutes_Enable
函数原型	void RTC_Alarm_B_Minutes_Enable (FunctionalState NewState)

功能描述	设置闹钟 B 分钟使能
输入参数 1	NewState:闹钟 B 分钟使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.42 函数 RTC_Alarm_B_Minutes_Config

表 20-47 函数 RTC_Alarm_B_Minutes_Config

函数名	RTC_Alarm_B_Minutes_Config
函数原型	void RTC_Alarm_B_Minutes_Config (uint32_t Minutes)
功能描述	配置闹钟中断分钟时间
输入参数 1	Minutes:闹钟中断分钟时间, 取值为 0-59
返回值	无
被调用函数	无

20.4.43 函数 RTC_Alarm_B_Seconds_Enable

表 20-48 函数 RTC_Alarm_B_Seconds_Enable

函数名	RTC_Alarm_B_Seconds_Enable
函数原型	void RTC_Alarm_B_Seconds_Enable (FunctionalState NewState)
功能描述	设置闹钟 B 秒使能
输入参数 1	NewState:闹钟 B 秒使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.44 函数 RTC_Alarm_B_Seconds_Config

表 20-49 函数 RTC_Alarm_B_Seconds_Config

函数名	RTC_Alarm_B_Seconds_Config
函数原型	void RTC_Alarm_B_Seconds_Config (uint32_t Seconds)
功能描述	配置闹钟中断秒时间
输入参数 1	Seconds:闹钟中断秒时间, 取值为 0-59
返回值	无
被调用函数	无

20.4.45 函数 RTC_Weekday_Config

表 20-50 函数 RTC_Weekday_Config

函数名	RTC_Weekday_Config
-----	--------------------

函数原型	void RTC_Weekday_Config (uint8_t Weekday)
功能描述	配置 RTC 时钟星期
输入参数 1	Weekday:RTC 时钟星期，取值为： RTC_WEEKDAY_MONDAY:星期一 RTC_WEEKDAY_TUESDAY:星期二 RTC_WEEKDAY_WEDNESDAY:星期三 RTC_WEEKDAY_THURSDAY:星期四 RTC_WEEKDAY_FRIDAY:星期五 RTC_WEEKDAY_SATURDAY:星期六 RTC_WEEKDAY_SUNDAY:星期天
返回值	无
被调用函数	无

20.4.46 函数 RTC_AMPM_Config

表 20-51 函数 RTC_AMPM_Config

函数名	RTC_AMPM_Config
函数原型	void RTC_AMPM_Config (uint32_t NewSelect)
功能描述	设置 RTC 时钟上午下午选择
输入参数 1	NewSelect:RTC 时钟上午下午选择，取值为： RTC_TIME_AM:选择上午时间 RTC_TIME_PM:选择下午时间
返回值	无
被调用函数	无

20.4.47 函数 RTC_Hours_Config

表 20-52 函数 RTC_Hours_Config

函数名	RTC_Hours_Config
函数原型	void RTC_Hours_Config (uint32_t Hour)
功能描述	配置 RTC 时钟时钟时间
输入参数 1	Hour:RTC 时钟时钟时间，取值匹配 12 小时制或 24 小时制
返回值	无
被调用函数	无

20.4.48 函数 RTC_Minutes_Config

表 20-53 函数 RTC_Minutes_Config

函数名	RTC_Minutes_Config
函数原型	void RTC_Minutes_Config (uint32_t Minutes)

功能描述	配置 RTC 时钟分钟时间
输入参数 1	Minutes:RTC 时钟分钟时间，取值为 0-59
返回值	无
被调用函数	无

20.4.49 函数 RTC_Seconds_Config

表 20-54 函数 RTC_Seconds_Config

函数名	RTC_Seconds_Config
函数原型	void RTC_Seconds_Config (uint32_t Seconds)
功能描述	配置 RTC 时钟秒钟时间
输入参数 1	Seconds:RTC 时钟秒钟时间，取值为 0-59
返回值	无
被调用函数	无

20.4.50 函数 RTC_Year_Config

表 20-55 函数 RTC_Year_Config

函数名	RTC_Year_Config
函数原型	void RTC_Year_Config (uint32_t Year)
功能描述	配置 RTC 时钟年份
输入参数 1	Year:RTC 时钟年份，取值为 0-99
返回值	无
被调用函数	无

20.4.51 函数 RTC_Month_Config

表 20-56 函数 RTC_Month_Config

函数名	RTC_Month_Config
函数原型	void RTC_Month_Config (uint32_t Month)
功能描述	配置 RTC 时钟月份
输入参数 1	Month:RTC 时钟月份，取值为: RTC_MONTH_JANUARY:1 月 RTC_MONTH_FEBRUARY:2 月 RTC_MONTH_MARCH:3 月 RTC_MONTH_APRIL:4 月 RTC_MONTH_MAY:5 月 RTC_MONTH_JUNE:6 月 RTC_MONTH_JULY:7 月 RTC_MONTH_AUGUST:8 月

	RTC_MONTH_SEPTMBER:9 月 RTC_MONTH_OCTOBER:10 月 RTC_MONTH_NOVEMBER:11 月 RTC_MONTH_DECEMBER:12 月
返回值	无
被调用函数	无

20.4.52 函数 RTC_Day_Config

表 20-57 函数 RTC_Day_Config

函数名	RTC_Day_Config
函数原型	void RTC_Day_Config (uint32_t Day)
功能描述	配置 RTC 时钟日期
输入参数 1	Day:RTC 时钟日期, 取值为 1-31
返回值	无
被调用函数	无

20.4.53 函数 RTC_Weekday_Backup_Config

表 20-58 函数 RTC_Weekday_Backup_Config

函数名	RTC_Weekday_Backup_Config
函数原型	void RTC_Weekday_Backup_Config (uint8_t Weekday)
功能描述	配置 RTC 时钟星期备份
输入参数 1	Weekday:RTC 时钟星期备份, 取值为: RTC_WEEKDAY_MONDAY:星期一 RTC_WEEKDAY_TUESDAY:星期二 RTC_WEEKDAY_WEDNESDAY:星期三 RTC_WEEKDAY_THURSDAY:星期四 RTC_WEEKDAY_FRIDAY:星期五 RTC_WEEKDAY_SATURDAY:星期六 RTC_WEEKDAY_SUNDAY:星期天
返回值	无
被调用函数	无

20.4.54 函数 RTC_AMPM_Backup_Config

表 20-59 函数 RTC_AMPM_Backup_Config

函数名	RTC_AMPM_Backup_Config
函数原型	void RTC_AMPM_Backup_Config (uint32_t NewSelect)
功能描述	设置 RTC 时钟上午下午选择备份

输入参数 1	NewSelect:RTC 时钟上午下午选择备份，取值为: RTC_TIME_AM:选择上午时间 RTC_TIME_PM:选择下午时间
返回值	无
被调用函数	无

20.4.55 函数 RTC_Hours_Backup_Config

表 20-60 函数 RTC_Hours_Backup_Config

函数名	RTC_Hours_Backup_Config
函数原型	void RTC_Hours_Backup_Config (uint32_t Hour)
功能描述	配置 RTC 时钟时间备份
输入参数 1	Hour:RTC 时钟时间备份，取值匹配 12 小时制或 24 小时制
返回值	无
被调用函数	无

20.4.56 函数 RTC_Minutes_Backup_Config

表 20-61 函数 RTC_Minutes_Backup_Config

函数名	RTC_Minutes_Backup_Config
函数原型	void RTC_Minutes_Backup_Config (uint32_t Minutes)
功能描述	配置 RTC 时钟分钟时间备份
输入参数 1	Minutes:RTC 时钟分钟时间备份，取值为 0-59
返回值	无
被调用函数	无

20.4.57 函数 RTC_Seconds_Backup_Config

表 20-62 函数 RTC_Seconds_Backup_Config

函数名	RTC_Seconds_Backup_Config
函数原型	void RTC_Seconds_Backup_Config (uint32_t Seconds)
功能描述	配置 RTC 时钟秒钟时间备份
输入参数 1	Seconds:RTC 时钟秒钟时间备份，取值为 0-59
返回值	无
被调用函数	无

20.4.58 函数 RTC_Year_Backup_Config

表 20-63 函数 RTC_Year_Backup_Config

函数名	RTC_Year_Backup_Config
函数原型	void RTC_Year_Backup_Config (uint32_t Year)
功能描述	配置 RTC 时钟年份备份
输入参数 1	Year:RTC 时钟年份备份, 取值为 0-99
返回值	无
被调用函数	无

20.4.59 函数 RTC_Month_Backup_Config

表 20-64 函数 RTC_Month_Backup_Config

函数名	RTC_Month_Backup_Config
函数原型	void RTC_Month_Backup_Config (uint32_t Month)
功能描述	配置 RTC 时钟月份备份
输入参数 1	Month:RTC 时钟月份备份, 取值为: RTC_MONTH_JANUARY:1 月 RTC_MONTH_FEBRUARY:2 月 RTC_MONTH_MARCH:3 月 RTC_MONTH_APRIL:4 月 RTC_MONTH_MAY:5 月 RTC_MONTH_JUNE:6 月 RTC_MONTH_JULY:7 月 RTC_MONTH_AUGUST:8 月 RTC_MONTH_SEPTMBER:9 月 RTC_MONTH_OCTOBER:10 月 RTC_MONTH_NOVEMBER:11 月 RTC_MONTH_DECEMBER:12 月
返回值	无
被调用函数	无

20.4.60 函数 RTC_Day_Backup_Config

表 20-65 函数 RTC_Day_Backup_Config

函数名	RTC_Day_Backup_Config
函数原型	void RTC_Day_Backup_Config (uint32_t Day)
功能描述	配置 RTC 时钟日期备份
输入参数 1	Day:RTC 时钟日期备份, 取值为 1-31
返回值	无
被调用函数	无

20.4.61 函数 RTC_Timer1_Config

表 20-66 函数 RTC_Timer1_Config

函数名	RTC_Timer1_Config
函数原型	void RTC_Timer1_Config (uint16_t Counter)
功能描述	配置 RTC 定时器 1 计数值
输入参数 1	Counter:RTC 定时器 1 计数值, 取值为 16 位数值
返回值	无
被调用函数	无

20.4.62 函数 RTC_Timer0_Config

表 20-67 函数 RTC_Timer0_Config

函数名	RTC_Timer0_Config
函数原型	void RTC_Timer0_Config (uint16_t Counter)
功能描述	配置 RTC 定时器 0 计数值
输入参数 1	Counter:RTC 定时器 0 计数值, 取值为 16 位数值
返回值	无
被调用函数	无

20.4.63 函数 RTC_Timer1_Enable

表 20-68 函数 RTC_Timer1_Enable

函数名	RTC_Timer1_Enable
函数原型	void RTC_Timer1_Enable (FunctionalState TimerEnable)
功能描述	设置 RTC 定时器 1 使能
输入参数 1	TimerEnable:RTC 定时器 1 使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.64 函数 RTC_Timer0_Enable

表 20-69 函数 RTC_Timer0_Enable

函数名	RTC_Timer0_Enable
函数原型	void RTC_Timer0_Enable (FunctionalState TimerEnable)
功能描述	设置 RTC 定时器 0 使能
输入参数 1	TimerEnable:RTC 定时器 0 使能状态, 取值为 TRUE 或 FALSE
返回值	无

被调用函数	无
-------	---

20.4.65 函数 RTC_Timer1_Source_Config

表 20-70 函数 RTC_Timer1_Source_Config

函数名	RTC_Timer1_Source_Config
函数原型	void RTC_Timer1_Source_Config (uint16_t ClockSource)
功能描述	配置定时器 1 时钟源选择
输入参数 1	<p>ClockSource:定时器 1 时钟源选择:</p> <p>RTC_TIMER_CLOCK_RTC_DIV_128: RTC 时钟源/128 约 1/256s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_512: RTC 时钟源/512 约 1/64s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_1024: RTC 时钟源/1024 约 1/32s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_2048: RTC 时钟源/2048 约 1/16s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_8192: RTC 时钟源/8192 约 1/4s</p> <p>RTC_TIMER_CLOCK_ONE_SECOND: 1s</p> <p>RTC_TIMER_CLOCK_ONE_MINUTE: 1min</p> <p>RTC_TIMER_CLOCK_ONE_HOUR: 1hour</p> <p>RTC_TIMER_CLOCK_CONNECTION: 计数器 TMR1 和 TMR0 级联</p>
返回值	无
被调用函数	无

20.4.66 函数 RTC_Timer0_Source_Config

表 20-71 函数 RTC_Timer0_Source_Config

函数名	RTC_Timer0_Source_Config
函数原型	void RTC_Timer0_Source_Config (uint16_t ClockSource)
功能描述	配置定时器 0 时钟源选择
输入参数 1	<p>ClockSource:定时器 1 时钟源选择:</p> <p>RTC_TIMER_CLOCK_RTC_DIV_128: RTC 时钟源/128 约 1/256s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_512: RTC 时钟源/512 约 1/64s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_1024: RTC 时钟源/1024 约 1/32s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_2048: RTC 时钟源/2048 约 1/16s</p> <p>RTC_TIMER_CLOCK_RTC_DIV_8192: RTC 时钟源/8192 约 1/4s</p> <p>RTC_TIMER_CLOCK_ONE_SECOND: 1s</p> <p>RTC_TIMER_CLOCK_ONE_MINUTE: 1min</p> <p>RTC_TIMER_CLOCK_ONE_HOUR: 1hour</p>
返回值	无
被调用函数	无

20.4.67 函数 RTC_Time_Stamp_INT_Enable

表 20-72 函数 RTC_Time_Stamp_INT_Enable

函数名	RTC_Time_Stamp_INT_Enable
函数原型	void RTC_Time_Stamp_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 时间戳中断使能
输入参数 1	NewState:RTC 时间戳中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.68 函数 RTC_Time_Stamp_OVERFLOW_INT_Enable

表 20-73 函数 RTC_Time_Stamp_OVERFLOW_INT_Enable

函数名	RTC_Time_Stamp_OVERFLOW_INT_Enable
函数原型	void RTC_Time_Stamp_OVERFLOW_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 时间戳溢出中断使能
输入参数 1	NewState:RTC 时间戳溢出中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.69 函数 RTC_Timer1_INT_Enable

表 20-74 函数 RTC_Timer1_INT_Enable

函数名	RTC_Timer1_INT_Enable
函数原型	void RTC_Timer1_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 定时器 1 中断使能
输入参数 1	NewState:RTC 定时器 1 中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.70 函数 RTC_Timer0_INT_Enable

表 20-75 函数 RTC_Timer0_INT_Enable

函数名	RTC_Timer0_INT_Enable
函数原型	void RTC_Timer0_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 定时器 0 中断使能
输入参数 1	NewState:RTC 定时器 0 中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.71 函数 RTC_Time_Tick_INT_Enable

表 20-76 函数 RTC_Time_Tick_INT_Enable

函数名	RTC_Time_Tick_INT_Enable
函数原型	void RTC_Time_Tick_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 时间节拍中断使能
输入参数 1	NewState:RTC 时间节拍中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.72 函数 RTC_Alarm_B_INT_Enable

表 20-77 函数 RTC_Alarm_B_INT_Enable

函数名	RTC_Alarm_B_INT_Enable
函数原型	void RTC_Alarm_B_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 闹钟 B 中断使能
输入参数 1	NewState:RTC 闹钟 B 中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.73 函数 RTC_Alarm_A_INT_Enable

表 20-78 函数 RTC_Alarm_A_INT_Enable

函数名	RTC_Alarm_A_INT_Enable
函数原型	void RTC_Alarm_A_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 闹钟 A 中断使能
输入参数 1	NewState:RTC 闹钟 A 中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.74 函数 RTC_Days_INT_Enable

表 20-79 函数 RTC_Days_INT_Enable

函数名	RTC_Days_INT_Enable
函数原型	void RTC_Days_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 日进程中中断使能
输入参数 1	NewState:RTC 日进程中中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.75 函数 RTC_Hours_INT_Enable

表 20-80 函数 RTC_Hours_INT_Enable

函数名	RTC_Hours_INT_Enable
函数原型	void RTC_Hours_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 小时进程中断使能
输入参数 1	NewState:RTC 小时进程中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.76 函数 RTC_Minutes_INT_Enable

表 20-81 函数 RTC_Minutes_INT_Enable

函数名	RTC_Minutes_INT_Enable
函数原型	void RTC_Minutes_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 分进程中断使能
输入参数 1	NewState:RTC 分进程中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.77 函数 RTC_Seconds_INT_Enable

表 20-82 函数 RTC_Seconds_INT_Enable

函数名	RTC_Seconds_INT_Enable
函数原型	void RTC_Seconds_INT_Enable (FunctionalState NewState)
功能描述	设置 RTC 秒进程中断使能
输入参数 1	NewState:RTC 秒进程中断使能状态, 取值为 TRUE 或 FALSE
返回值	无
被调用函数	无

20.4.78 函数 RTC_Get_Time_Stamp_INT_Flag

表 20-83 函数 RTC_Get_Time_Stamp_INT_Flag

函数名	RTC_Get_Time_Stamp_INT_Flag
函数原型	FlagStatus RTC_Get_Time_Stamp_INT_Flag (void)
功能描述	获取时间戳中断标志
输入参数 1	无
返回值	1:产生了时间戳中断 0:未产生时间戳中断
被调用函数	无

20.4.79 函数 RTC_Get_Time_Stamp_Overflow_INT_Flag

表 20-84 函数 RTC_Get_Time_Stamp_Overflow_INT_Flag

函数名	RTC_Get_Time_Stamp_Overflow_INT_Flag
函数原型	FlagStatus RTC_Get_Time_Stamp_Overflow_INT_Flag (void)
功能描述	获取时间戳溢出中断标志
输入参数 1	无
返回值	1:产生了时间戳溢出中断 0:未产生时间戳溢出中断
被调用函数	无

20.4.80 函数 RTC_Get_Timer1_INT_Flag

表 20-85 函数 RTC_Get_Timer1_INT_Flag

函数名	RTC_Get_Timer1_INT_Flag
函数原型	FlagStatus RTC_Get_Timer1_INT_Flag (void)
功能描述	获取 RTC 定时器 1 中断标志
输入参数 1	无
返回值	1:产生了 RTC 定时器 1 中断 0:未产生 RTC 定时器 1 中断
被调用函数	无

20.4.81 函数 RTC_Get_Timer0_INT_Flag

表 20-86 函数 RTC_Get_Timer0_INT_Flag

函数名	RTC_Get_Timer0_INT_Flag
函数原型	FlagStatus RTC_Get_Timer0_INT_Flag (void)
功能描述	获取 RTC 定时器 0 中断标志
输入参数 1	无
返回值	1:产生了 RTC 定时器 0 中断 0:未产生 RTC 定时器 0 中断
被调用函数	无

20.4.82 函数 RTC_Get_Timer0_INT_Flag

表 20-87 函数 RTC_Get_Timer0_INT_Flag

函数名	RTC_Get_Time_Tick_INT_Flag
函数原型	FlagStatus RTC_Get_Time_Tick_INT_Flag (void)
功能描述	获取 RTC 时间节拍中断标志

输入参数 1	无
返回值	1:产生了时间节拍中断 0:未产生时间节拍中断
被调用函数	无

20.4.83 函数 RTC_Get_Alarm_B_INT_Flag

表 20-88 函数 RTC_Get_Alarm_B_INT_Flag

函数名	RTC_Get_Alarm_B_INT_Flag
函数原型	FlagStatus RTC_Get_Alarm_B_INT_Flag (void)
功能描述	获取 RTC 闹钟 B 中断标志
输入参数 1	无
返回值	1:产生了闹钟 B 中断 0:未产生闹钟 B 中断
被调用函数	无

20.4.84 函数 RTC_Get_Alarm_A_INT_Flag

表 20-89 函数 RTC_Get_Alarm_A_INT_Flag

函数名	RTC_Get_Alarm_A_INT_Flag
函数原型	FlagStatus RTC_Get_Alarm_A_INT_Flag (void)
功能描述	获取 RTC 闹钟 A 中断标志
输入参数 1	无
返回值	1:产生了闹钟 A 中断 0:未产生闹钟 B 中断
被调用函数	无

20.4.85 函数 RTC_Get_Days_INT_Flag

表 20-90 函数 RTC_Get_Days_INT_Flag

函数名	RTC_Get_Days_INT_Flag
函数原型	FlagStatus RTC_Get_Days_INT_Flag (void)
功能描述	获取 RTC 日进程中中断标志
输入参数 1	无
返回值	1:产生了日进程中中断 0:未产生日进程中中断
被调用函数	无

20.4.86 函数 RTC_Get_Hours_INT_Flag

表 20-91 函数 RTC_Get_Hours_INT_Flag

函数名	RTC_Get_Hours_INT_Flag
函数原型	FlagStatus RTC_Get_Hours_INT_Flag (void)
功能描述	获取 RTC 小时进程中断标志
输入参数 1	无
返回值	1:产生了小时进程中断 0:未产生小时进程中断
被调用函数	无

20.4.87 函数 RTC_Get_Minutes_INT_Flag

表 20-92 函数 RTC_Get_Minutes_INT_Flag

函数名	RTC_Get_Minutes_INT_Flag
函数原型	FlagStatus RTC_Get_Minutes_INT_Flag (void)
功能描述	获取 RTC 分进程中断标志
输入参数 1	无
返回值	1:产生了分进程中断 0:未产生分进程中断
被调用函数	无

20.4.88 函数 RTC_Get_Seconds_INT_Flag

表 20-93 函数 RTC_Get_Seconds_INT_Flag

函数名	RTC_Get_Seconds_INT_Flag
函数原型	FlagStatus RTC_Get_Seconds_INT_Flag (void)
功能描述	获取 RTC 秒进程中断标志
输入参数 1	无
返回值	1:产生了秒进程中断 0:未产生秒进程中断
被调用函数	无

20.4.89 函数 RTC_Clear_Time_Stamp_INT_Flag

表 20-94 函数 RTC_Clear_Time_Stamp_INT_Flag

函数名	RTC_Clear_Time_Stamp_INT_Flag
函数原型	void RTC_Clear_Time_Stamp_INT_Flag (void)
功能描述	清零 RTC 时间戳中断标志
输入参数 1	无

返回值	无
被调用函数	无

20.4.90 函数 RTC_Clear_Time_Stamp_Overflow_INT_Flag

表 20-95 函数 RTC_Clear_Time_Stamp_Overflow_INT_Flag

函数名	RTC_Clear_Time_Stamp_Overflow_INT_Flag
函数原型	void RTC_Clear_Time_Stamp_Overflow_INT_Flag (void)
功能描述	清零 RTC 时间戳溢出中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.91 函数 RTC_Clear_Timer1_INT_Flag

表 20-96 函数 RTC_Clear_Timer1_INT_Flag

函数名	RTC_Clear_Timer1_INT_Flag
函数原型	void RTC_Clear_Timer1_INT_Flag (void)
功能描述	清零 RTC 定时器 1 中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.92 函数 RTC_Clear_Timer0_INT_Flag

表 20-97 函数 RTC_Clear_Timer0_INT_Flag

函数名	RTC_Clear_Timer0_INT_Flag
函数原型	void RTC_Clear_Timer0_INT_Flag (void)
功能描述	清零 RTC 定时器 0 中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.93 函数 RTC_Clear_Time_Tick_INT_Flag

表 20-98 函数 RTC_Clear_Time_Tick_INT_Flag

函数名	RTC_Clear_Time_Tick_INT_Flag
函数原型	void RTC_Clear_Time_Tick_INT_Flag (void)
功能描述	清零时间节拍中断标志

输入参数 1	无
返回值	无
被调用函数	无

20.4.94 函数 RTC_Clear_Alarm_B_INT_Flag

表 20-99 函数 RTC_Clear_Alarm_B_INT_Flag

函数名	RTC_Clear_Alarm_B_INT_Flag
函数原型	void RTC_Clear_Alarm_B_INT_Flag (void)
功能描述	清零闹钟 B 中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.95 函数 RTC_Clear_Alarm_A_INT_Flag

表 20-100 函数 RTC_Clear_Alarm_A_INT_Flag

函数名	RTC_Clear_Alarm_A_INT_Flag
函数原型	void RTC_Clear_Alarm_A_INT_Flag (void)
功能描述	清零闹钟 A 中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.96 函数 RTC_Clear_Days_INT_Flag

表 20-101 函数 RTC_Clear_Days_INT_Flag

函数名	RTC_Clear_Days_INT_Flag
函数原型	void RTC_Clear_Days_INT_Flag (void)
功能描述	清零 RTC 日进程中中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.97 函数 RTC_Clear_Hours_INT_Flag

表 20-102 函数 RTC_Clear_Hours_INT_Flag

函数名	RTC_Clear_Hours_INT_Flag
函数原型	void RTC_Clear_Hours_INT_Flag (void)

功能描述	清零 RTC 小时进程中中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.98 函数 RTC_Clear_Minutes_INT_Flag

表 20-103 函数 RTC_Clear_Minutes_INT_Flag

函数名	RTC_Clear_Minutes_INT_Flag
函数原型	void RTC_Clear_Minutes_INT_Flag (void)
功能描述	清零 RTC 分进程中中断标志
输入参数 1	无
返回值	无
被调用函数	无

20.4.99 函数 RTC_Clear_Seconds_INT_Flag

表 20-104 函数 RTC_Clear_Seconds_INT_Flag

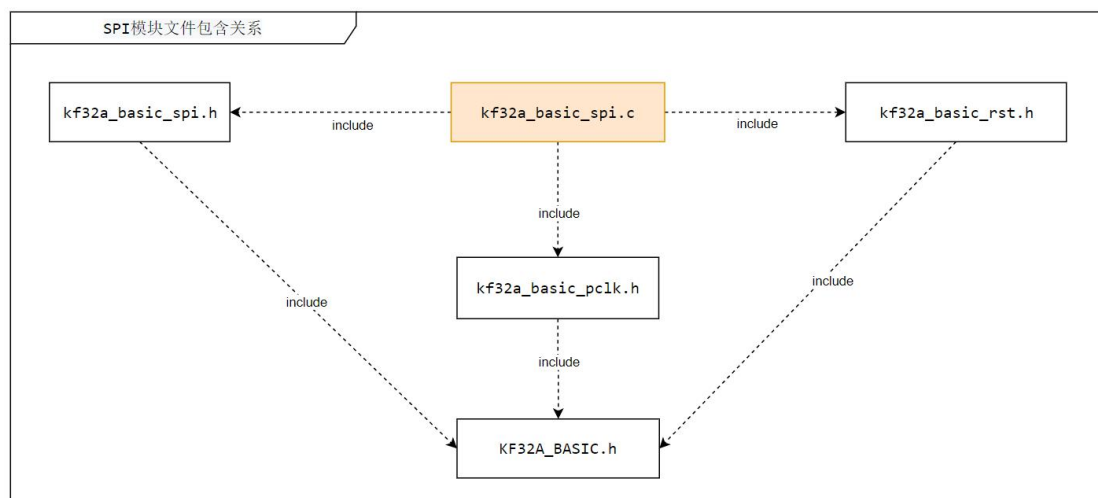
函数名	RTC_Clear_Seconds_INT_Flag
函数原型	void RTC_Clear_Seconds_INT_Flag (void)
功能描述	清零 RTC 秒进程中中断标志
输入参数 1	无
返回值	无
被调用函数	无

21 串行外设接口（SPI）

SPI 模块可配置为支持 SPI 协议或者 I2S 协议。SPI 模块默认工作在 SPI 方式，可通过软件将其切换到 I2S 模式。在 I2S 模式下，原则上数据传输为全双工模式，主机和从机同时收发数据，但实际情况下通常只有一个方向上的数据是有意义的。

21.1 文件引用关系

图 21-1 SPI 模块文件包含关系



21.2 SPI 寄存器结构

SPI 寄存器结构，SPI_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct SPI_MemMap {
    volatile uint32_t    BRGR;
    volatile uint32_t    CTLR;
    volatile uint32_t    BUFR;
    volatile uint32_t    STR;
} SPI_SFRmap;
    
```

表 21-1 SPI 寄存器结构说明

寄存器	描述
BRGR	SPI 控制寄存器,偏移:0x0
CTLR	SPI 波特率寄存器,偏移:0x4
BUFR	SPI 数据寄存器,偏移:0x8
STR	SPI 状态寄存器,偏移:0xC

SPI 协议信息结构体，定义于文件 kf32a_basic_spi.h 中，如下：

```
typedef struct
{
    uint32_t    m_Mode
    uint32_t    m_Clock;
    uint32_t    m_FirstBit;
    uint32_t    m_CKP;
    uint32_t    m_CKE;
    uint32_t    m_DataSize;
    uint16_t    m_BaudRate;
} SPI_InitTypeDef;
```

表 21-2 SPI 协议信息结构体说明

配置信息	描述
m_Mode	SPI 模式选择，取值为宏“SPI 模式”中的一个。
m_Clock	SPI 工作时钟选择寄存器，取值为宏“SPI 工作时钟”中的一个。
m_FirstBit	SPI 数据方向选择位.取值为宏“SPI 数据方向”中的一个。
m_CKP	时钟极性选择位.取值为宏“SPI 时钟极性”中的一个。
m_CKE	SPI 时钟边沿选择位.取值为宏“SPI 时钟边沿”中的一个。
m_DataSize	SPI 位模式选择，取值为宏“SPI 位模式”中的一个。
m_BaudRate	SPI 波特率选择，取值为 0~0xFFFF。

I2S 协议信息结构体，定义于文件 kf32a_basic_spi.h 中，如下：

```
typedef struct
{
    uint32_t    m_Mode;
    uint32_t    m_Standard;
    uint32_t    m_PCM;
    uint32_t    m_Clock;
    uint32_t    m_CKP;
    uint8_t     m_Prescaler;
} I2S_InitTypeDef;
```

表 21-3 I2S 协议信息结构体说明

配置信息	描述
m_Mode	I2S 模式选择，取值为宏“I2S 模式”中的一个。
m_Standard	I2S 标准选择，取值为宏“I2S 标准”中的一个。
m_PCM	PCM 帧同步，取值为宏“PCM 帧同步”中的一个。
m_Clock	I2S 模式的时钟分频选择，取值为宏“I2S 模式的时钟分频”中的一个。
m_CKP	PCM 模式时钟极性选择，取值为宏“PCM 模式时钟极性选择”中的一个。
m_Prescaler	I2S 预分频，取值为 0~0xFF。

21.3 SPI 宏定义

SPI 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，SPI 其他相关宏定义详见文件 kf32a_basic_spi.h 及函数参数描述。

21.4 SPI 库函数

表 21-4 SPI 固件库函数列表

序号	函数名	描述
1	SPI_Reset	SPI 外设复位。
2	SPI_Configuration	SPI 模式初始化配置。
3	I2S_Configuration	I2S 模式初始化配置。
4	SPI_Struct_Init	初始化 SPI 协议信息结构体。
5	I2S_Struct_Init	初始化 I2S 协议信息结构体。
6	SPI_Cmd	控制 SPI 模块总使能位。
7	I2S_Mode_Select	控制 I2S 选择位。
8	SPI_I2S_ReceiveData	获取 SPI_I2S 接收数据。
9	SPI_I2S_SendData8	配置 SPI_I2S 发送 32 位数据。
10	SPI_I2S_SendData32	配置 SPI_I2S 发送 8 位数据。
11	SPI_BaudRate_Config	配置 SPI 波特率寄存器。
12	I2S_DIV_Config	配置 I2S 预分频寄存器。
13	SPI_MODE_Config	配置 SPI 同步串行端口模式选择位。
14	SPI_CLK_Config	配置 SPI 工作时钟选择寄存器。
15	SPI_Data_Direction_Config	设置 SPI 数据方向选择位。
16	SPI_Clock_Polarity_Config	设置 SPI 时钟极性选择位。
17	SPI_Clock_Edge_Config	设置 SPI 时钟边沿选择位。
18	SPI_BIT_SELECT_Config	配置 SPI 位模式选择。
19	SPI_I2S_MODE_Config	配置 I2S 模式设置。
20	SPI_I2S_STANDARD_Config	配置 I2S 标准选择。
21	SPI_PCM_Config	设置 I2S PCM 帧同步。
22	SPI_CHLEN_Config	设置 I2S 模式的时钟分频选择。
23	SPI_PCM_CLOCK_Polarity_Config	设置 PCM 模式时钟极性选择。
24	SPI_MAIN_CLOCK_OUT_Enable	设置主设备时钟输出使能。
25	SPI_Receive_Overflow_INT_Enable	设置 SPI 接收溢出中断使能。
26	SPI_Transmit_Overflow_INT_Enable	设置 SPI 发送溢出中断使能。
27	SPI_RNEIE_INT_Enable	设置 SPI RBUF 不为空中断使能。
28	SPI_TNEIE_INT_Enable	设置 SPI TBUF 为空中断使能。
29	SPI_Receive_DMA_INT_Enable	设置 SPI 接收 DMA 中断使能。
30	SPI_Transmit_DMA_INT_Enable	设置 SPI 发送 DMA 中断使能。
31	SPI_Transmit_CHSIDE_INT_Enable	设置 SPI 发送声道选择。

32	SPI_Get_BUSY_Flag	获取 SPI 忙状态。
33	SPI_Get_Receive_Overflow_Flag	获取 SPI 接收溢出中断标志状态。
34	SPI_Get_Transmit_Overflow_Flag	获取 SPI 产生发送溢出标志状态。
35	SPI_Get_Receive_Buf_Flag	获取 SPI 接收 BUF 未空状态。
36	SPI_Get_Transmit_Buf_Flag	获取 SPI 发送 BUF 未空状态。
37	SPI_Clear_Receive_Overflow_INT_Flag	清零 SPI 接收溢出中断标志。
38	SPI_Clear_Transmit_Overflow_INT_Flag	清零 SPI 发送溢出中断标志。

21.4.1 函数 SPI_Reset

表 21-5 函数 SPI_Reset

函数名	SPI_Reset
函数原型	void SPI_Reset(SPI_SFRmap* SPIx)
功能描述	SPI 外设复位。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	无
被调用函数 1	void RST_CTL1_Peripheral_Reset_Enable (uint32_t RST_CTL1Periph, FunctionalState NewState);
被调用函数 2	void PCLK_CTL1_Peripheral_Clock_Enable (uint32_t PCLK_CTL1_bit, FunctionalState NewState);
被调用函数 3	void RST_CTL3_Peripheral_Reset_Enable (uint32_t RST_CTL3_bit, FunctionalState NewState);
被调用函数 4	void PCLK_CTL3_Peripheral_Clock_Enable (uint32_t PCLK_CTL3_bit, FunctionalState NewState);

21.4.2 函数 SPI_Configuration

表 21-6 函数 SPI_Configuration

函数名	SPI_Configuration
函数原型	void SPI_Configuration(SPI_SFRmap* SPIx, SPI_InitTypeDef* spiInitStruct)
功能描述	SPI 模式初始化配置。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	SpiInitStruct: SPI 协议信息结构体指针。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

21.4.3 函数 I2S_Configuration

表 21-7 函数 I2S_Configuration

函数名	I2S_Configuration
函数原型	void I2S_Configuration(SPI_SFRmap* SPIx, I2S_InitTypeDef* i2sInitStruct)
功能描述	I2S 模式初始化配置。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	i2sInitStruct: I2S 协议信息结构体指针。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

21.4.4 函数 SPI_Struct_Init

表 21-8 函数 SPI_Struct_Init

函数名	SPI_Struct_Init
函数原型	void SPI_Struct_Init(SPI_InitTypeDef* SPI_InitStruct)
功能描述	初始化 SPI 协议信息结构体。
输入参数 1	SPI_InitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

21.4.5 函数 I2S_Struct_Init

表 21-9 函数 I2S_Struct_Init

函数名	I2S_Struct_Init
函数原型	void I2S_Struct_Init(I2S_InitTypeDef* I2S_InitStruct)
功能描述	初始化 I2S 协议信息结构体。
输入参数 1	I2S_InitStruct: 指向待初始化的结构体指针。
返回值	无
被调用函数	无

21.4.6 函数 SPI_Cmd

表 21-10 函数 SPI_Cmd

函数名	SPI_Cmd
函数原型	void SPI_Cmd (SPI_SFRmap* SPIx, FunctionalState NewState)
功能描述	控制 SPI 模块总使能位。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI 使能位配置信息, 取值为 TRUE 或 FALSE。

返回值	无
被调用函数	无

21.4.7 函数 I2S_Mode_Select

表 21-11 函数 I2S_Mode_Select

函数名	I2S_Mode_Select
函数原型	void I2S_Mode_Select(SPI_SFRmap* SPIx, FunctionalState NewState)
功能描述	控制 I2S 选择位。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: I2S 选择位配置信息, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21.4.8 函数 SPI_I2S_ReceiveData

表 21-12 函数 SPI_I2S_ReceiveData

函数名	SPI_I2S_ReceiveData
函数原型	uint32_t SPI_I2S_ReceiveData (SPI_SFRmap* SPIx)
功能描述	获取 SPI_I2S 接收数据。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	32 位接收数据。
被调用函数	无

21.4.9 函数 SPI_I2S_SendData32

表 21-13 函数 SPI_I2S_SendData32

函数名	SPI_I2S_SendData32
函数原型	void SPI_I2S_SendData32(SPI_SFRmap* SPIx, uint32_t Data)
功能描述	配置 SPI_I2S 发送 32 位数据。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	Data: 待发送 32 位数据。
返回值	无
被调用函数	无

21.4.10 函数 SPI_I2S_SendData8

表 21-14 函数 SPI_I2S_SendData8

函数名	SPI_I2S_SendData8
-----	-------------------

函数原型	void SPI_I2S_SendData8(SPI_SFRmap* SPIx, uint8_t Data)
功能描述	配置 SPI_I2S 发送 8 位数据。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	Data: 待发送 8 位数据。
返回值	无
被调用函数	无

21.4.11 函数 SPI_BaudRate_Config

表 21-15 函数 SPI_BaudRate_Config

函数名	SPI_BaudRate_Config
函数原型	void SPI_BaudRate_Config (SPI_SFRmap* SPIx, uint16_t BAUDRATE)
功能描述	配置 SPI 波特率寄存器。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	BAUDRATE: 波特率选择, 取值为 0~65535。
返回值	无
被调用函数	无

21.4.12 函数 I2S_DIV_Config

表 21-16 函数 I2S_DIV_Config

函数名	I2S_DIV_Config
函数原型	void I2S_DIV_Config (SPI_SFRmap* SPIx, uint8_t DIV)
功能描述	配置 I2S 预分频寄存器。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	DIV: I2S 预分频选择, 取值为 0~255。
返回值	无
被调用函数	无

21.4.13 函数 SPI_MODE_Config

表 21-17 函数 SPI_MODE_Config

函数名	SPI_MODE_Config
函数原型	void SPI_MODE_Config (SPI_SFRmap* SPIx, uint32_t MODE)
功能描述	配置 SPI 同步串行端口模式选择位。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	MODE: 模式选择, 取值范围为: SPI_MODE_MASTER_CLKDIV4: SPI 主控模式, 时钟 = 工作时钟/4 SPI_MODE_MASTER_CLKDIV16: SPI 主控模式, 时钟 = 工作时钟/16 SPI_MODE_MASTER_CLKDIV64: SPI 主控模式, 时钟 = 工作时钟/64

	SPI_MODE_MASTER_T2DIV2: SPI 主控模式, 时钟 = TIMER2/2 SPI_MODE_SLAVE: SPI 从动模式, 时钟 = SCK 引脚。使能 SS 引脚控制
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

21.4.14 函数 SPI_CLK_Config

表 21-18 函数 SPI_CLK_Config

函数名	SPI_CLK_Config
函数原型	void SPI_CLK_Config (SPI_SFRmap* SPIx, uint32_t ClockSource)
功能描述	配置 SPI 工作时钟选择寄存器。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	ClockSource: 模式选择, 取值范围为: SPI_CLK_SCLK :选用 SCLK 为 SPI 工作时钟 SPI_CLK_HFCLK :选用 HFCLK 为 SPI 工作时钟 SPI_CLK_LFCLK :选用 LFCLK 为 SPI 工作时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

21.4.15 函数 SPI_Data_Direction_Config

表 21-19 函数 SPI_Data_Direction_Config

函数名	SPI_Data_Direction_Config
函数原型	void SPI_Data_Direction_Config(SPI_SFRmap* SPIx, uint32_t DataDirection)
功能描述	设置 SPI 数据方向选择位。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	DataDirection: SPI 数据方向, 取值为: SPI_FIRSTBIT_MSB: (最高位) 在前, SPI_FIRSTBIT_LSB: (最低位) 在前。
返回值	无
被调用函数	无

21.4.16 函数 SPI_Clock_Polarity_Config

表 21-20 函数 SPI_Clock_Polarity_Config

函数名	SPI_Clock_Polarity_Config
函数原型	void SPI_Clock_Polarity_Config (SPI_SFRmap* SPIx, uint32_t Polarity)
功能描述	设置 SPI 时钟极性选择位。

输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	Polarity: SPI 时钟极性, 取值为: SPI_CKP_LOW:空闲状态时, 时钟为低电平; SPI_CKP_HIGH:空闲状态时, 时钟为高电平。
返回值	无
被调用函数	无

21.4.17 函数 SPI_Clock_Edge_Config

表 21-21 函数 SPI_Clock_Edge_Config

函数名	SPI_Clock_Edge_Config
函数原型	void SPI_Clock_Edge_Config (SPI_SFRmap* SPIx,uint32_t ClockEdge)
功能描述	设置 SPI 时钟边沿选择位。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	ClockEdge: SPI 时钟边沿, 取值为: SPI_CKE_2EDGE:在第 2 个时钟沿 (包括上升, 下降) 发送数据, SPI_CKE_1EDGE:在第 1 个时钟沿 (包括上升, 下降) 发送数据。
返回值	无
被调用函数	无

21.4.18 函数 SPI_BIT_SELECT_Config

表 21-22 函数 SPI_BIT_SELECT_Config

函数名	SPI_BIT_SELECT_Config
函数原型	void SPI_BIT_SELECT_Config(SPI_SFRmap* SPIx,uint32_t DataSize)
功能描述	配置 SPI 位模式选择。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	DataSize: 模式选择, 取值范围为: SPI_DATASIZE_8BITS :SPI 使用 8 位模式 (不支持 I2S 模式) SPI_DATASIZE_16BITS :SPI 使用 16 位模式 SPI_DATASIZE_32BITS :SPI 使用 32 位模式
返回值	无
被调用函数	无

21.4.19 函数 SPI_I2S_MODE_Config

表 21-23 函数 SPI_I2S_MODE_Config

函数名	SPI_I2S_MODE_Config
函数原型	void SPI_I2S_MODE_Config (SPI_SFRmap* SPIx,uint32_t Mode)
功能描述	配置 I2S 模式设置。

输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	Mode: 模式选择, 取值范围为: I2S_MODE_SLAVE_TX :从设备发送模式 I2S_MODE_SLAVE_RX :从设备接收模式 I2S_MODE_MASTER_TX :主设备发送模式 I2S_MODE_MASTER_RX :主设备接收模式
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

21.4.20 函数 SPI_I2S_STANDARD_Config

表 21-24 函数 SPI_I2S_STANDARD_Config

函数名	SPI_I2S_STANDARD_Config
函数原型	void SPI_I2S_STANDARD_Config(SPI_SFRmap* SPIx,uint32_t Standard)
功能描述	配置 I2S 标准选择。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	Standard: 模式选择, 取值范围为: I2S_STANDARD_PHILLIPS :I2S 飞利浦标准 I2S_STANDARD_MSB :低字节对齐标准 (右对齐) I2S_STANDARD_LSB :高字节对齐标准 (左对齐) I2S_STANDARD_PCM :PCM 标准
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

21.4.21 函数 SPI_PCM_Config

表 21-25 函数 SPI_PCM_Config

函数名	SPI_PCM_Config
函数原型	void SPI_PCM_Config (SPI_SFRmap* SPIx,uint32_t NewState)
功能描述	设置 I2S PCM 帧同步。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: PCM 帧同步配置, 取值为: I2S_PCM_LONG: 长帧同步, I2S_PCM_SHORT:短帧同步。
返回值	无
被调用函数	无

21.4.22 函数 SPI_CHLEN_Config

表 21-26 函数 SPI_CHLEN_Config

函数名	SPI_CHLEN_Config
函数原型	void SPI_CHLEN_Config (SPI_SFRmap* SPIx,uint32_t NewState)
功能描述	设置 I2S 模式的时钟分频选择。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: 时钟分频配置, 取值为: I2S_CLK_BAUDRATE:正常波特率, I2S_CLK_BAUDRATEDIV2:正常波特率/2。
返回值	无
被调用函数	无

21.4.23 函数 SPI_PCM_CLOCK_Polarity_Config

表 21-27 函数 SPI_PCM_CLOCK_Polarity_Config

函数名	SPI_PCM_CLOCK_Polarity_Config
函数原型	void SPI_PCM_CLOCK_Polarity_Config(SPI_SFRmap* SPIx,uint32_t NewState)
功能描述	设置 PCM 模式时钟极性选择。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: PCM 模式时钟极性, 取值为: I2S_CKP_HIGH: PCM 数据在时钟下降沿发送, I2S_CKP_LOW: PCM 数据在时钟上升沿发送。
返回值	无
被调用函数	无

21.4.24 函数 SPI_MAIN_CLOCK_OUT_Enable

表 21-28 函数 SPI_MAIN_CLOCK_OUT_Enable

函数名	SPI_MAIN_CLOCK_OUT_Enable
函数原型	void SPI_MAIN_CLOCK_OUT_Enable (SPI_SFRmap* SPIx,FunctionalState NewState)
功能描述	设置主设备时钟输出使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: 主设备时钟输出使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21.4.25 函数 SPI_Receive_Overflow_INT_Enable

表 21-29 函数 SPI_Receive_Overflow_INT_Enable

函数名	SPI_Receive_Overflow_INT_Enable
函数原型	void SPI_Receive_Overflow_INT_Enable (SPI_SFRmap*SPIx,FunctionalState NewState)
功能描述	设置 SPI 接收溢出中断使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI 接收溢出中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21.4.26 函数 SPI_Transmit_Overflow_INT_Enable

表 21-30 函数 SPI_Transmit_Overflow_INT_Enable

函数名	SPI_Transmit_Overflow_INT_Enable
函数原型	void SPI_Transmit_Overflow_INT_Enable(SPI_SFRmap*SPIx,FunctionalState NewState)
功能描述	设置 SPI 发送溢出中断使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI 发送溢出中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21.4.27 函数 SPI_RNEIE_INT_Enable

表 21-31 函数 SPI_RNEIE_INT_Enable

函数名	SPI_RNEIE_INT_Enable
函数原型	void SPI_RNEIE_INT_Enable (SPI_SFRmap* SPIx, FunctionalState NewState)
功能描述	设置 SPI RBUF 不为空中断使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI RBUF 不为空中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21.4.28 函数 SPI_TNEIE_INT_Enable

表 21-32 函数 SPI_TNEIE_INT_Enable

函数名	SPI_TNEIE_INT_Enable
函数原型	void SPI_TNEIE_INT_Enable (SPI_SFRmap* SPIx,FunctionalState NewState)

功能描述	设置 SPI TBUF 为空中断使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI TBUF 为空中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21. 4. 29 函数 SPI_Receive_DMA_INT_Enable

表 21-33 函数 SPI_Receive_DMA_INT_Enable

函数名	SPI_Receive_DMA_INT_Enable
函数原型	void SPI_Receive_DMA_INT_Enable (SPI_SFRmap* SPIx,FunctionalState NewState)
功能描述	设置 SPI 接收 DMA 中断使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI 接收 DMA 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21. 4. 30 函数 SPI_Transmit_DMA_INT_Enable

表 21-34 函数 SPI_Transmit_DMA_INT_Enable

函数名	SPI_Transmit_DMA_INT_Enable
函数原型	void SPI_Transmit_DMA_INT_Enable (SPI_SFRmap* SPIx,FunctionalState NewState)
功能描述	设置 SPI 发送 DMA 中断使能。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI 发送 DMA 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

21. 4. 31 函数 SPI_Transmit_CHSIDE_INT_Enable

表 21-35 函数 SPI_Transmit_CHSIDE_INT_Enable

函数名	SPI_Transmit_CHSIDE_INT_Enable
函数原型	void SPI_Transmit_CHSIDE_INT_Enable (SPI_SFRmap* SPIx,FunctionalState NewState)
功能描述	设置 SPI 发送声道选择。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
输入参数 2	NewState: SPI 发送声道选择, 取值为 TRUE 或 FALSE。
返回值	无

被调用函数	无
-------	---

21.4.32 函数 SPI_Get_BUSY_Flag

表 21-36 函数 SPI_Get_BUSY_Flag

函数名	SPI_Get_BUSY_Flag
函数原型	FlagStatus SPI_Get_BUSY_Flag (SPI_SFRmap* SPIx)
功能描述	获取 SPI 忙状态。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	1: SPI 正在发送数据; 0: SPI 没有在发送数据。
被调用函数	无

21.4.33 函数 SPI_Get_Receive_Overflow_Flag

表 21-37 函数 SPI_Get_Receive_Overflow_Flag

函数名	SPI_Get_Receive_Overflow_Flag
函数原型	FlagStatus SPI_Get_Receive_Overflow_Flag (SPI_SFRmap* SPIx)
功能描述	获取 SPI 接收溢出中断标志状态。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	1: SPI 接收溢出; 0: SPI 没有接收溢出。
被调用函数	无

21.4.34 函数 SPI_Get_Transmit_Overflow_Flag

表 21-38 函数 SPI_Get_Transmit_Overflow_Flag

函数名	SPI_Get_Transmit_Overflow_Flag
函数原型	FlagStatus SPI_Get_Transmit_Overflow_Flag (SPI_SFRmap* SPIx)
功能描述	获取 SPI 产生发送溢出标志状态。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	1: SPI 产生发送溢出; 0: SPI 没有产生发送溢出。
被调用函数	无

21.4.35 函数 SPI_Get_Receive_Buf_Flag

表 21-39 函数 SPI_Get_Receive_Buf_Flag

函数名	SPI_Get_Receive_Buf_Flag
-----	--------------------------

函数原型	FlagStatus SPI_Get_Receive_Buf_Flag (SPI_SFRmap* SPIx)
功能描述	获取 SPI 接收 BUF 未空状态。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	1: SPI 接收 BUF 不为空; 0: SPI 接收 BUF 为空。
被调用函数	无

21.4.36 函数 SPI_Get_Transmit_Buf_Flag

表 21-40 函数 SPI_Get_Transmit_Buf_Flag

函数名	SPI_Get_Transmit_Buf_Flag
函数原型	FlagStatus SPI_Get_Transmit_Buf_Flag (SPI_SFRmap* SPIx)
功能描述	获取 SPI 发送 BUF 未空状态。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	1: SPI 发送 BUF 不为空; 0: SPI 发送 BUF 为空。
被调用函数	无

21.4.37 函数 SPI_Clear_Receive_Overflow_INT_Flag

表 21-41 函数 SPI_Clear_Receive_Overflow_INT_Flag

函数名	SPI_Clear_Receive_Overflow_INT_Flag
函数原型	void SPI_Clear_Receive_Overflow_INT_Flag (SPI_SFRmap* SPIx)
功能描述	清零 SPI 接收溢出中断标志。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	无
被调用函数	无

21.4.38 函数 SPI_Clear_Transmit_Overflow_INT_Flag

表 21-42 函数 SPI_Clear_Transmit_Overflow_INT_Flag

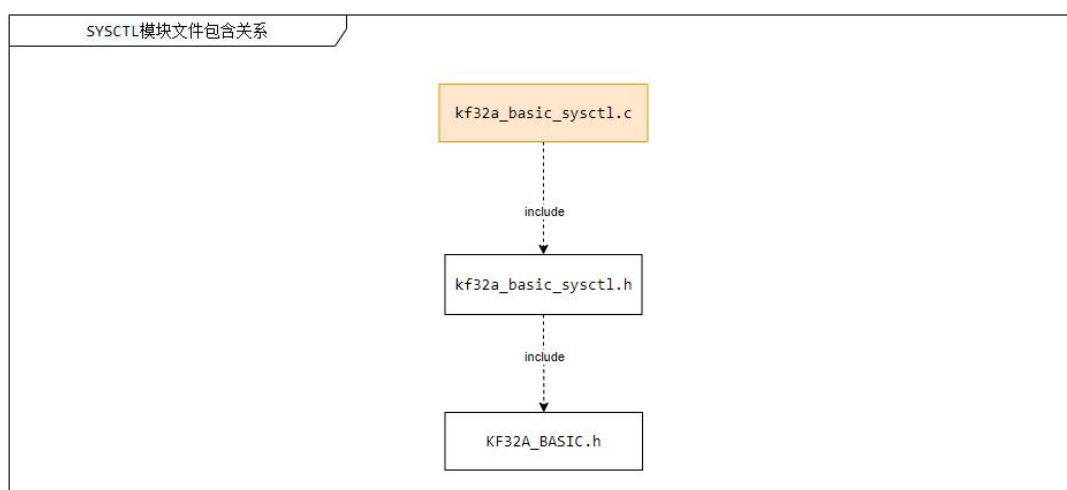
函数名	SPI_Clear_Transmit_Overflow_INT_Flag
函数原型	void SPI_Clear_Transmit_Overflow_INT_Flag (SPI_SFRmap* SPIx)
功能描述	清零 SPI 发送溢出中断标志。
输入参数 1	SPIx: 指向 SPI 内存结构的指针, 取值为 SPI0_SFR~SPI3_SFR。
返回值	无
被调用函数	无

22 系统控制（SYSCTL）

通过系统控制模块（SYSCTL）可以使用运算结果标志位 N/V/C/Z；可以选择 MSP 或是 PSP 作为堆栈指针；可以获得当前堆栈的对齐情况；可以实现软件产生系统复位；可以实现系统休眠的控制；可以设置中断向量表的首地址以此实现中断向量表的重映射；可以设置当前芯片的 RAM 空间的大小。

22.1 文件引用关系

图 22-1 SYSCTL 模块文件包含关系



22.2 SYSCTL 寄存器结构

SYSCTL 寄存器结构，SYSCTL_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct SYSCTL_MemMap {
    volatile uint32_t    PSW;
    volatile uint32_t    MCTL;
    volatile uint32_t    ARCTL;
    volatile uint32_t    VECTOFF;
    uint32_t             RESEVRVE1;
    volatile uint32_t    RAMSPA;
    volatile uint32_t    MEMCTL;
}SYSCTL_SFRmap;
  
```

表 22-1 SYSCTL 寄存器结构说明

寄存器	描述
PSW	程序状态寄存器，偏移:0x00
MCTL	系统模式控制寄存器，偏移:0x04

ARCTL	应用复位控制, 偏移:0x08
VECTOFF	中断向量表重映射控制寄存器, 偏移:0x0C
RESEVRVE1	保留, 偏移:0x10
RAMSPA	RAM 空间指示寄存器, 偏移:0x14
MEMCTL	程序空间控制寄存器, 偏移:0x18

22.3 SYSCTL 宏定义

SYSCTL 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, SYSCTL 其他相关宏定义详见文件 kf32a_basic_sysctl.h 及函数参数描述。

22.4 SYSCTL 库函数

表 22-2 SYSCTL 固件库函数列表

序号	函数名	描述
1	SYSCTL_Get_V_Flag	获取状态字 V 位溢出标志
2	SYSCTL_Get_C_Flag	获取状态字 C 位进位或借位标志
3	SYSCTL_Get_Z_Flag	获取状态字 Z 位零值标志
4	SYSCTL_Get_N_Flag	获取状态字 N 位负值标志
5	SYSCTL_Set_V_Flag	设置状态字 V 位溢出标志
6	SYSCTL_Set_C_Flag	设置状态字 C 位溢出标志
7	SYSCTL_Set_Z_Flag	设置状态字 Z 位溢出标志
8	SYSCTL_Set_N_Flag	设置状态字 N 位负值标志
9	SYSCTL_Sleep_On_Exit_Enable	设置中断处理返回进入休眠模式使能
10	SYSCTL_Deep_Sleep_Enable	设置深度休眠模式使能
11	SYSCTL_Interrupt_Awake_Enable	设置唤醒模式选择
12	SYSCTL_Stack_Align_State	获取当前中断自动堆栈的对齐方式
13	SYSCTL_Super_User_Config	设置超级用户控制
14	SYSCTL_Stack_Pointer_State	获取当前有效堆栈指针状态标志
15	SYSCTL_Stack_Pointer_Config	设置当前有效堆栈指针
16	SYSCTL_Exception_Reset_Enable	设置异常活动信息复位请求
17	SYSCTL_System_Reset_Enable	设置系统复位请求
18	SYSCTL_Vector_Offset_Config	中断向量表重映射地址配置
19	SYSCTL_Ram_Space_Config	配置 RAM 空间长度, 指示 RAM 空间结束地址
20	SYSCTL_Flash_Start_Remap_Config	配置 FLASH 开始 512 字节空间映射控制

22.4.1 函数 SYSCTL_Get_V_Flag

表 22-3 函数 SYSCTL_Get_V_Flag

函数名	SYSCTL_Get_V_Flag
函数原型	FlagStatus SYSCTL_Get_V_Flag (void)
功能描述	获取状态字 V 位溢出标志
输入参数 1	无
返回值	1:溢出; 0:非溢出
被调用函数	无

22.4.2 函数 SYSCTL_Get_C_Flag

表 22-4 函数 SYSCTL_Get_C_Flag

函数名	SYSCTL_Get_C_Flag
函数原型	FlagStatus SYSCTL_Get_C_Flag (void)
功能描述	获取状态字 C 位进位或借位标志
输入参数 1	无
返回值	无
被调用函数	1:加法有进位或减法无借位; 0:加法无进位或减法有借位

22.4.3 函数 SYSCTL_Get_Z_Flag

表 22-5 函数 SYSCTL_Get_Z_Flag

函数名	SYSCTL_Get_Z_Flag
函数原型	FlagStatus SYSCTL_Get_Z_Flag (void)
功能描述	获取状态字 Z 位零值标志
输入参数 1	无
返回值	1:零值; 0:非零值
被调用函数	无

22.4.4 函数 SYSCTL_Get_N_Flag

表 22-6 函数 SYSCTL_Get_N_Flag

函数名	SYSCTL_Get_N_Flag
函数原型	FlagStatus SYSCTL_Get_N_Flag (void)
功能描述	获取状态字 N 位负值标志
输入参数 1	无
返回值	1:负值; 0:非负值
被调用函数	无

22.4.5 函数 SYSCTL_Set_V_Flag

表 22-7 函数 SYSCTL_Set_V_Flag

函数名	SYSCTL_Set_V_Flag
函数原型	void SYSCTL_Set_V_Flag (FunctionalState NewState)
功能描述	设置状态字 V 位溢出标志
输入参数 1	NewState: 状态字 V 位溢出标志, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.6 函数 SYSCTL_Set_C_Flag

表 22-8 函数 SYSCTL_Set_C_Flag

函数名	SYSCTL_Set_C_Flag
函数原型	void SYSCTL_Set_C_Flag (FunctionalState NewState)
功能描述	设置状态字 C 位溢出标志
输入参数 1	NewState: 状态字 C 位进位或借位标志, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.7 函数 SYSCTL_Set_Z_Flag

表 22-9 函数 SYSCTL_Set_Z_Flag

函数名	SYSCTL_Set_Z_Flag
函数原型	void SYSCTL_Set_Z_Flag (FunctionalState NewState)
功能描述	设置状态字 Z 位溢出标志
输入参数 1	NewState: 状态字 Z 位进位或借位标志, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.8 函数 SYSCTL_Set_N_Flag

表 22-10 函数 SYSCTL_Set_N_Flag

函数名	SYSCTL_Set_N_Flag
函数原型	void SYSCTL_Set_N_Flag (FunctionalState NewState)
功能描述	设置状态字 N 位负值标志
输入参数 1	NewState: 状态字 N 位负值或非负值标志, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.9 函数 SYSCTL_Sleep_On_Exit_Enable

表 22-11 函数 SYSCTL_Sleep_On_Exit_Enable

函数名	SYSCTL_Sleep_On_Exit_Enable
函数原型	void SYSCTL_Sleep_On_Exit_Enable (FunctionalState NewState)
功能描述	设置中断处理返回进入休眠模式使能
输入参数 1	NewState: 中断处理返回进入休眠模式使能状态, 取值为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.10 函数 SYSCTL_Deep_Sleep_Enable

表 22-12 函数 SYSCTL_Deep_Sleep_Enable

函数名	SYSCTL_Deep_Sleep_Enable
函数原型	void SYSCTL_Deep_Sleep_Enable (FunctionalState NewState)
功能描述	设置深度休眠模式使能
输入参数 1	NewState: 深度休眠模式使能状态, 取值为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.11 函数 SYSCTL_Interrupt_Awake_Enable

表 22-13 函数 SYSCTL_Interrupt_Awake_Enable

函数名	SYSCTL_Interrupt_Awake_Enable
函数原型	void SYSCTL_Interrupt_Awake_Enable (FunctionalState NewState)
功能描述	设置唤醒模式选择
输入参数 1	NewState: 任意中断唤醒 CPU 使能状态, 取值为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.12 函数 SYSCTL_Stack_Align_State

表 22-14 函数 SYSCTL_Stack_Align_State

函数名	SYSCTL_Stack_Align_State
函数原型	FlagStatus SYSCTL_Stack_Align_State (void)
功能描述	获取当前中断自动堆栈的对齐方式
输入参数 1	无
返回值	1:双字对齐使能; 0:双字对齐未使能
被调用函数	无

22.4.13 函数 SYSCTL_Super_User_Config

表 22-15 函数 SYSCTL_Super_User_Config

函数名	SYSCTL_Super_User_Config
函数原型	void SYSCTL_Super_User_Config (FunctionalState NewState)
功能描述	设置超级用户控制
输入参数 1	NewState: 选择超级用户标志, 取值为: TRUE 或 FALSE
返回值	无
被调用函数	无

22.4.14 函数 SYSCTL_Stack_Pointer_State

表 22-16 函数 SYSCTL_Stack_Pointer_State

函数名	SYSCTL_Stack_Pointer_State
函数原型	FlagStatus SYSCTL_Stack_Pointer_State (void)
功能描述	获取当前有效堆栈指针状态标志
输入参数 1	无
返回值	1:PSP 是当前的堆栈指针; 0:MSP 是当前的堆栈指针
被调用函数	无

22.4.15 函数 SYSCTL_Stack_Pointer_Config

表 22-17 函数 SYSCTL_Stack_Pointer_Config

函数名	SYSCTL_Stack_Pointer_Config
函数原型	void SYSCTL_Stack_Pointer_Config (uint32_t PresentSP)
功能描述	设置当前有效堆栈指针
输入参数 1	PresentSP: 当前有效堆栈指针, 取值为: SYSCTL_SP_IS_MSP: MSP 是当前的堆栈指针 SYSCTL_SP_IS_PSP: PSP 是当前的堆栈指针
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

22.4.16 函数 SYSCTL_Exception_Reset_Enable

表 22-18 函数 SYSCTL_Exception_Reset_Enable

函数名	SYSCTL_Exception_Reset_Enable
函数原型	void SYSCTL_Exception_Reset_Enable (FunctionalState NewState)

功能描述	设置异常活动信息复位请求
输入参数 1	NewState: 异常活动信息复位请求, 取值为: TRUE 或 FALSE
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

22. 4. 17 函数 SYSCTL_System_Reset_Enable

表 22-19 函数 SYSCTL_System_Reset_Enable

函数名	SYSCTL_System_Reset_Enable
函数原型	void SYSCTL_System_Reset_Enable (FunctionalState NewState)
功能描述	设置系统复位请求
输入参数 1	NewState: 系统复位请求, 取值为: TRUE 或 FALSE
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

22. 4. 18 函数 SYSCTL_Vector_Offset_Config

表 22-20 函数 SYSCTL_Vector_Offset_Config

函数名	SYSCTL_Vector_Offset_Config
函数原型	void SYSCTL_Vector_Offset_Config (uint32_t VectorOffset)
功能描述	中断向量表重映射地址配置
输入参数 1	VectorOffset: 中断向量表重映射地址, 取值 32 位数据, 低两位硬件忽略
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

22. 4. 19 函数 SYSCTL_Ram_Space_Config

表 22-21 函数 SYSCTL_Ram_Space_Config

函数名	SYSCTL_Ram_Space_Config
函数原型	void SYSCTL_Ram_Space_Config (uint32_t RamSpace)
功能描述	配置 RAM 空间长度, 指示 RAM 空间结束地址
输入参数 1	RamSpace: RAM 空间长度, 取值为 26 位有效数值
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

22.4.20 函数 SYSCTL_Flash_Start_Remap_Config

表 22-22 函数 SYSCTL_Flash_Start_Remap_Config

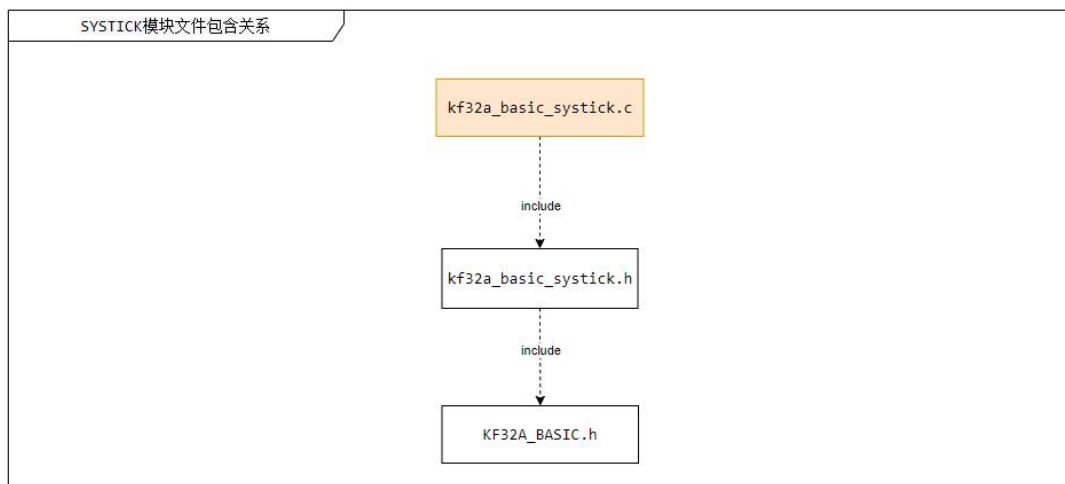
函数名	SYSCTL_Flash_Start_Remap_Config
函数原型	void SYSCTL_Flash_Start_Remap_Config (uint32_t MemCtl)
功能描述	配置 FLASH 开始 512 字节空间映射控制
输入参数 1	MemCtl: 映射控制, 取值范围为: SYSCTL_FLASH_REMAP_FOR_ROM: 为 ROM 的映射 SYSCTL_FLASH_REMAP_FOR_RAM: 为 RAM 的映射 SYSCTL_FLASH_REMAP_FOR_FLASH: 为 FLASH 的映射 SYSCTL_FLASH_REMAP_STOP_CPU: 停止 CPU 运行
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

23 节拍定时器（SYSTICK）

KungFu32 内核提供了一个 24 位的系统节拍定时器（System Tick Timer）。系统节拍定时器可为系统提供可编程时长的周期性中断，能工作在休眠模式下。

23.1 文件引用关系

图 23-1 SYSTICK 模块文件包含关系



23.2 SYSTICK 寄存器结构

SYSTICK 寄存器结构，SYSTICK_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct SYSTICK_MemMap
{
    volatile uint32_t    CTL;
    volatile uint32_t    RELOAD;
    volatile uint32_t    CV;
    volatile uint32_t    CALI;
}SYSTICK_SFRmap;
  
```

表 23-1 SYSTICK 寄存器结构说明

寄存器	描述
CTL	系统节拍定时器控制和状态寄存器，偏移:0x00
RELOAD	系统节拍定时器重载值寄存器，偏移:0x04
CV	系统节拍定时器当前值寄存器，偏移:0x08
CALI	系统节拍定时器校验寄存器，偏移:0x0C

SYSTICK 配置信息结构体，SYSTICK_SFRmap，定义于文件 kf32a_basic_systick.h 中，

如下:

```
typedef struct
{
    uint32_t    m_Period;
    uint32_t    m_Clock;
    uint32_t    m_SysTickINT;
} SYSTICK_InitTypeDef;
```

表 23-2 SYSTICK 配置信息结构体说明

配置信息	描述
m_Period	节拍定时器重加载寄存器的值，取值 24 位数据
m_Clock	节拍定时器的时钟源选择，取值为宏“SYSTICK 定时器时钟源”中的一个
m_SysTickINT	定时器 SysTick 中断使能配置，取值为 TRUE 或 FALSE

23.3 SYSTICK 宏定义

SYSTICK 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，SYSTICK 其他相关宏定义详见文件 kf32a_basic_systick.h 及函数参数描述。

23.4 SYSTICK 库函数

表 23-3 SYSTICK 固件库函数列表

序号	函数名	描述
1	SYSTICK_Configuration	节拍定时器配置
2	SYSTICK_Cmd	节拍定时器启动控制使能
3	SYSTICK_Clock_Config	节拍定时器时钟源配置
4	SYSTICK_Systick_INT_Enable	节拍定时器中断使能配置
5	SYSTICK_Get_Count_Zero_Flag	读取节拍定时器计数零标志
6	SYSTICK_Reload_Config	节拍定时器重加载寄存器配置
7	SYSTICK_Counter_Update	节拍定时器计数器配置，该函数会清零 COUNTZERO 位，并将 STRELOAD 值能加载到计数器寄存器中
8	SYSTICK_Get_Reload	读取节拍定时器重加载寄存器
9	SYSTICK_Get_Counter	读取节拍定时器计数器

23.4.1 函数 SYSTICK_Configuration

表 23-4 函数 SYSTICK_Configuration

函数名	SYSTICK_Configuration
函数原型	Void SYSTICK_Configuration (uint32_t Clock, uint32_t SysTickINT,

	uint32_t Reload)
功能描述	设置 IO 口状态锁存使能
输入参数 1	Clock:系统节拍定时器时钟源选择, 取值范围为: SYSTICK_SYS_CLOCK_DIV_1: SCLK 作为时钟 SYSTICK_SYS_CLOCK_DIV_2: SCLK/2 作为时钟
输入参数 2	SysTickINT: SysTick 中断使能控制, 取值范围为: TRUE 或 FALSE
输入参数 3	Reload: 系统节拍定时器重加载值, 节拍定时器的周期值为重加载值加 1, 取值为 24 位数据
返回值	无
被调用函数	无

23.4.2 函数 SYSTICK_Cmd

表 23-5 函数 SYSTICK_Cmd

函数名	SYSTICK_Cmd
函数原型	void SYSTICK_Cmd (FunctionalState NewState)
功能描述	节拍定时器启动控制使能
输入参数 1	NewState: 节拍定时器使能控制, 取值范围为: TRUE 或 FALSE
返回值	无
被调用函数	无

23.4.3 函数 SYSTICK_Clock_Config

表 23-6 函数 SYSTICK_Clock_Config

函数名	SYSTICK_Clock_Config
函数原型	void SYSTICK_Clock_Config (uint32_t SysClock)
功能描述	节拍定时器时钟源配置
输入参数 1	SysClock: 系统节拍定时器时钟源选择, 取值范围为: SYSTICK_SYS_CLOCK_DIV_1: SCLK 作为时钟 SYSTICK_SYS_CLOCK_DIV_2: SCLK/2 作为时钟
返回值	无
被调用函数	无

23.4.4 函数 SYSTICK_Systick_INT_Enable

表 23-7 函数 SYSTICK_Systick_INT_Enable

函数名	SYSTICK_Systick_INT_Enable
函数原型	void SYSTICK_Systick_INT_Enable (uint32_t SysClock)
功能描述	节拍定时器 SysTick 中断使能配置
输入参数 1	SysClock: 系统节拍定时器 SysTick 中断使能状态, 取值范围为: TRUE 或

	FALSE
返回值	无
被调用函数	无

23.4.5 函数 SYSTICK_Get_Count_Zero_Flag

表 23-8 函数 SYSTICK_Get_Count_Zero_Flag

函数名	SYSTICK_Get_Count_Zero_Flag
函数原型	FlagStatus SYSTICK_Get_Count_Zero_Flag (void)
功能描述	读取节拍定时器计数零标志
输入参数 1	无
返回值	节拍定时器计数到零的标志，0：计数未到零，1：计数到零
被调用函数	无

23.4.6 函数 SYSTICK_Reload_Config

表 23-9 函数 SYSTICK_Reload_Config

函数名	SYSTICK_Reload_Config
函数原型	void SYSTICK_Reload_Config (uint32_t Reload)
功能描述	节拍定时器重加载寄存器配置
输入参数 1	Reload: 系统节拍定时器重加载值，节拍定时器的周期值为重加载值加 1。 取值为 24 位数据
返回值	无
被调用函数	无

23.4.7 函数 SYSTICK_Counter_Update

表 23-10 函数 SYSTICK_Counter_Update

函数名	SYSTICK_Counter_Update
函数原型	void SYSTICK_Counter_Update (void)
功能描述	节拍定时器计数器配置，该函数会清零 COUNTZERO 位，并将 STRELOAD 值能加载到计数器寄存器中
输入参数 1	无
返回值	无
被调用函数	无

23.4.8 函数 SYSTICK_Get_Reload

表 23-11 函数 SYSTICK_Get_Reload

函数名	SYSTICK_Get_Reload
函数原型	void SYSTICK_Get_Reload (void)
功能描述	读取系统节拍定时器重加载寄存器
输入参数 1	无
返回值	重加载寄存器的值，24 位有效数据
被调用函数	无

23.4.9 函数 SYSTICK_Get_Counter

表 23-12 函数 SYSTICK_Get_Counter

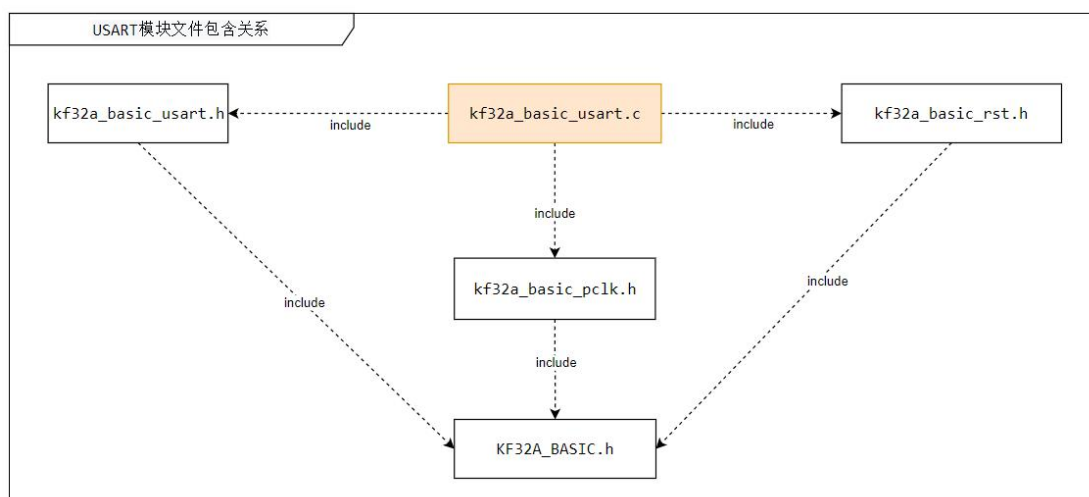
函数名	SYSTICK_Get_Counter
函数原型	void SYSTICK_Get_Counter (void)
功能描述	读取系统节拍定时器计数器
输入参数 1	无
返回值	当前计数值，24 位有效数据
被调用函数	无

24 通用同步/异步收发器（USART）

KungFu32 内核提供了多个独立的 USART（Universal Synchronous /Asynchronous Receive & Transmit）。这是一个串口通信的 I/O 外设，也可作为串行通信接口。它可被配置为与个人计算机等外设通信的全双工异步系统。也可以被配置为与外设或其它单片机通信的半双工同步系统，与之通信的单片机通常不具有产生波特率的内部时钟，它需要主控同步器件提供外部时钟信号。

24.1 文件引用关系

图 24-1 USART 寄存器结构说明



24.2 USART 寄存器结构

USART 寄存器结构，USART_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct USART_MemMap {
    volatile uint32_t    CTLR;
    volatile uint32_t    BRGR;
    volatile uint32_t    STR;
    union {
        volatile uint32_t    TBUFR;
        volatile const uint32_t    RBUFR;
    };
    volatile uint32_t    U7816R;
    volatile uint32_t    IER;
    volatile uint32_t    ADM;
}USART_SFRmap;

```

表 24-1 USART 寄存器结构说明

寄存器	描述
CTLR	USART 控制寄存器, 偏移:0x0
BRGR	USART 波特率寄存器, 偏移:0x4
STR	USART 状态寄存器, 偏移:0x8
TBUFR	USART 发送数据寄存器, 偏移:0xC
RBUFR	USART 接收数据寄存器, 偏移:0xC
U7816R	7816 控制寄存器, 偏移:0x10
IER	USART 中断使能寄存器, 偏移:0x14
ADM	地址匹配设置寄存器, 偏移:0x18

USART 配置信息结构体, USART_InitTypeDef, 定义于文件 kf32a_basic_usart.h 中, 如下:

```
typedef struct
{
    uint32_t    m_Mode;
    uint32_t    m_HalfDuplexClkSource;
    uint32_t    m_TransferDir;
    uint32_t    m_WordLength;
    uint32_t    m_StopBits;
    uint32_t    m_Bit9SEL;
    uint32_t    m_Parity;
    uint32_t    m_ReceiveOrder;
    uint32_t    m_TransmitOrder;
    uint32_t    m_BRAutoDetect;
    uint32_t    m_HardwareFlowControl;
    uint16_t    m_BaudRateBRCKS;
    uint16_t    m_BaudRateInteger;
    uint16_t    m_BaudRateNumerator;
    uint16_t    m_BaudRateDenominator;
} USART_InitTypeDef;
```

表 24-2 USART 配置信息结构体说明

配置信息	描述
m_Mode	USART 模式配置,取值为宏“USART 模式选择”中的一个
m_HalfDuplexClkSource	时钟源选择,取值为宏“USART 时钟源选择”中的一个。
m_TransferDir	USART 传输方向,取值为宏“USART 传输方向”中的一个。
m_WordLength	USART 字长配置取值为宏“USART 字长”中的一个
m_StopBits	USART 停止位配置 取值为宏“USART 停止位”中的一个
m_Bit9SEL	USART 发送数据第 9 位数据选择.取值为宏“USART 发送数据第 9 位数据选择”中的一个
m_Parity	USART 奇偶校验位配置取值为宏“USART 奇偶校验位”中的一

	个
m_ReceiveOrder	USART 接收次序设置为宏“接收次序选择”中的一个
m_TransmitOrder	USART 发送次序设置为宏“发送次序选择”中的一个
m_BRAutoDetect	USART 自动波特率模式设置为宏“自动波特率检测使能位”中的一个
m_HardwareFlowControl	USART 硬件流控制配置取值为宏“USART 硬件流”中的一个
m_BaudRateBRCKS	USART 波特率发生器时钟选择取值为宏“USART 时钟配置”中的一个
m_BaudRateInteger	USART 波特率整数部分，取值为 0~65535。
m_BaudRateNumerator	USART 波特率小数分子部分，取值为 0~0xF。
m_BaudRateDenominator	USART 波特率小数分母部分，取值为 0~0xF。

USART_7816 配置信息结构体，U7816R_InitTypeDef，定义于文件 kf32a_basic_usart.h 中，如下：

```
typedef struct
{
    uint32_t    m_ErrorSignal;
    uint32_t    m_PassagewaySelect;
    uint32_t    m_TransmitRepeat;
    uint32_t    m_ReceiveRepeat;
    FunctionalState    m_Clkout;
    uint8_t     m_ClkDiv;
    uint8_t     m_Egt;
} U7816R_InitTypeDef;
```

表 24-3 USART_7816 配置信息结构体说明

配置信息	描述
m_ErrorSignal	USART 错误宽度，取值为宏“USART 错误宽度选择配置”中的一个
m_PassagewaySelect	USART 数据通道. 取值为宏“USART 数据通道选择”中的一个
m_TransmitRepeat	USART 奇偶校验出错时重发送的最大次数，取值为宏“USART 奇偶校验出错时重发送的最大次数”中的一个
m_ReceiveRepeat	USART 奇偶校验出错时重接收的最大次数，取值为宏“USART 奇偶校验出错时重接收的最大次数”中的一个
m_Clkout	USB 端点输入使能，取值为 TRUE 或 FALSE。
m_ClkDiv	7816 工作时钟和引脚输出时钟控制，取值为 0~0xFF。
m_Egt	发送时插入的 EGT，取值为 0~0xFF。

24.3 USART 宏定义

USART 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h, USART 其他相关宏定义详见文件 kf32a_basic_usart.h 及函数参数描述。

24.4 USART 库函数

表 24-4 USART 固件库函数列表

序号	函数名	描述
1	USART_Reset	USART 外设复位
2	USART_Configuration	配置 USART 外设初始化。
3	USART_U7816R_Configuration	7816 外设配置。
4	USART_Struct_Init	初始化 USART 配置信息结构体。
5	USART_U7816R_Struct_Init	初始化 USART_7816 配置信息结构体。
6	USART_Cmd	配置 USART 使能。
7	USART_BaudRate_Clock_Config	配置 USART 波特率发生器时钟选择。
8	USART_HalfDuplex_ClockPolarity_Config	设置 USART 半双工同步时钟极性选择。
9	USART_Transmit_Order_Config	设置 USART 发送次序选择。
10	USART_Receive_Order_Config	设置 USART 接收次序选择。
11	USART_Infrare_Detector_Voltage_Config	设置红外调制高电平输出极性。
12	USART_WeakUP_Enable	设置 USART 唤醒使能。
13	USART_Clock_Source_Config	设置 USART 时钟源选择。
14	USART_Address_Detection_Enable	设置 USART 地址检测使能位。
15	USART_Auto_BaudRate_Detection_Enable	设置 USART 自动波特率检测使能位。
16	USART_Send_Blank_Enable	设置 USART 发送间隔字符使能。
17	USART_SYN_Choice_Config	设置 USART 串行通信模式选择位。
18	USART_Transmit_Data_Enable	设置 USART 发送使能位。
19	USART_Receive_Data_Enable	设置 USART 接收使能位。
20	USART_STOP_Word_Config	设置 USART 停止位长度选择位。
21	USART_Transmit_9Word_Select_Config	设置 USART 发送数据第 9 位数据选择。
22	USART_Parity_Select_Config	设置 USART 奇偶校验选择。
23	USART_9Data_Enable	设置 USART 第 9 位数据使能位。
24	USART_CTS_Enable	设置 USART CTS 使能位。
25	USART_RTS_Enable	设置 USART RTS 使能位。
26	USART_Infrare_Detector_Enable	设置 USART 红外调制使能位。
27	USART_Singlet_Line_Mode_Enable	设置 USART 单线模式使能位。
28	USART_BaudRate_Integer_Config	配置 USART USARTDIV 的整数部分。
29	USART_BaudRate_Decimal1_Config	配置 USART 小数波特率分子。
30	USART_BaudRate_Decimal2_Config	配置 USART 小数波特率分母。
31	USART_SendData	USART 发送数据。

32	USART_TransmitData	USART 发送字节,并等待发送器和缓冲器空。
33	USART_ReceiveData	USART 接收数据。
34	USART_Address_Match_Config	配置 USARTx 地址匹配功能设置位。
35	USART_Send_Idle_Frame_Enable	配置 USART 全双工模式时发送空闲帧使能。
36	USART_Receive_Idle_Frame_Config	配置 USART 接收空闲帧中断标志产生模式。
37	USART_7816_Cmd	设置 USART 7816 模式使能控制。
38	USART_7816_CLKOUT_Enable	设置 USART 7816 时钟输出使能。
39	USART_7816_Error_Signal_Config	配置 USART 7816 error signal 宽度选择。
40	USART_Passageway_Select_Config	设置 USART 数据通道选择。
41	USART_BGT_Config	设置 USART 控制接收到发送之间是否插入 BGT。
42	USART_Transmit_Repeat_Enable	设置 USART 重发使能控制。
43	USART_Receive_Repeat_Enable	设置 USART 重收使能控制。
44	USART_Transmit_Repeat_Times_Config	配置 USART 奇偶校验出错时重发送的最大次数。
45	USART_Receive_Repeat_Times_Config	配置 USART 奇偶校验出错时重接收的最大次数。
46	USART_7816_CLKDIV_Config	配置 USART 7816 工作时钟和引脚输出时钟控制。
47	USART_7816_EGT_Config	配置 USART 7816 发送时插入的 EGT(extra guard time) (单位 etu)。
48	USART_7816_Resend_Mode_Select	配置 USART 7816 重发模式选择。
49	USART_Receive_Overflow_INT_Enable	设置 USART 接收溢出中断使能。
50	USART_Parity_ERROR_INT_Enable	设置 USART 奇偶校验错误中断使能。
51	USART_Frame_ERROE_INT_Enable	设置 USART 帧错误中断使能。
52	USART_Blank_INT_Enable	设置 USART 间隔符号中断使能。
53	USART_Auto_BaudRate_TimeOver_INT_Enable	设置 USART 自动波特率超时中断使能。
54	USART_WeakUP_INT_Enable	设置 USART 自动唤醒中断使能。
55	USART_Transmit_ERROR_INT_Enable	设置 USART7816 发送错误中断使能。
56	USART_Receive_ERROR_INT_Enable	设置 USART7816 接收错误中断使能。
57	USART_CTS_INT_Enable	设置 USART CTS 中断使能。
58	USART_RDR_INT_Enable	设置 USART RDR 中断使能。

59	USART_TFE_INT_Enable	设置 USART TFE 中断使能。
60	USART_TXE_INT_Enable	设置 USART TXE 中断使能
61	USART_Receive_DMA_INT_Enable	设置 USART 接收数据 DMA 中断使能。
62	USART_Transmit_DMA_INT_Enable	设置 USART 发送数据 DMA 中断使能。
63	USART_IDLE_INT_Enable	设置 USART IDLEIF 中断使能。
64	USART_UADM_INT_Enable	设置 USART UADMIF 中断使能。
65	USART_Get_Receive_Overflow_Flag	获取 USART 接收溢出中断标志状态
66	USART_Get_Parity_ERROR_Flag	获取 USART 奇偶校验错误中断标志状态。
67	USART_Get_Frame_ERROR_Flag	获取 USART 帧错误中断标志状态。
68	USART_Get_Blank_Flag	获取 USART 间隔符中断标志状态。
69	USART_Get_Auto_Baudrate_TimeOver_Flag	获取 USART 自动波特率超时中断标志状态。
70	USART_Get_WeakUP_Flag	获取 USART 自动唤醒中断标志状态。
71	USART_Get_7816Transmit_ERROR_Flag	获取 USART 7816 发送错误中断标志状态。
72	USART_Get_7816Receive_ERROR_Flag	获取 USART 7816 接收错误中断标志状态。
73	USART_Get_CTS_Flag	获取 USART CTS 中断标志状态。
74	USART_Get_Receive_BUFR_Ready_Flag	获取 USART 数据就绪中断标志状态。
75	USART_Get_Transmit_BUFR_Empty_Flag	获取 USART 发送 BUF 为空中断标志状态。
76	USART_Get_Transmitter_Empty_Flag	获取 USART 发射器为空中断标志状态。
77	USART_Get_Receive_Frame_Idel_Flag	获取 USART 接收空闲帧中断标志
78	USART_Clear_Receive_Overflow_INT_Flag	清零 USART 接收溢出中断标志。
79	USART_Clear_Parity_ERROR_INT_Flag	清零 USART 奇偶校验错误位。
80	USART_Clear_Frame_ERROR_INT_Flag	清零 USART 帧错误位。
81	USART_Clear_Blank_INT_Flag	清零 USART 间隔符位。
82	USART_Clear_Auto_BaudRate_TimeOver_INT_Flag	清零 USART 自动波特率超时位。
83	USART_Clear_WeakUP_INT_Flag	清零 USART 自动唤醒中断位。
84	USART_Clear_Transmit_ERROR_INT_Flag	清零 USART 发送错误中断位。
85	USART_Clear_Receive_ERROR_INT_Flag	清零 USART 接收错误中断位。
86	USART_Clear_CTS_INT_Flag	清零 USART CTS 中断位。
87	USART_Clear_UADM_INT_Flag	清零 USART UADM 中断位。

88	USART_Clear_IDLE_INT_Flag	清零 USART IDLE 中断位。
89	USART_Clear_Receive_BUFR_INT_Flag	清零 USART 接收 BUF 中断位。
90	USART_Clear_Transmit_BUFR_INT_Flag	清零 USART 发送 BUF 中断位。
91	USART_Get_WUEN_Flag	获取 USART 唤醒使能位状态。
92	USART_Get_Auto_BaudRate_Detection_Flag	读取 USART 自动波特率检测使能位。
93	USART_RESHD_Enable	设置 USART RESHD 位。

24.4.1 函数 USART_Reset

表 24-5 函数 USART_Reset

函数名	USART_Reset
函数原型	void USART_Reset (USART_SFRmap USARTx)
功能描述	USART 外设复位
输入参数 1	USARTx:指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
返回值	无
被调用函数 1	void RST_CTL2_Peripheral_Reset_Enable (uint32_t RST_CTL2_bit, FunctionalState NewState)
被调用函数 2	void PCLK_CTL2_Peripheral_Clock_Enable (uint32_t PCLK_CTL2_bit, FunctionalState NewState)
被调用函数 3	void RST_CTL1_Peripheral_Reset_Enable (uint32_t RST_CTL1_bit, FunctionalState NewState)
被调用函数 4	void PCLK_CTL1_Peripheral_Clock_Enable (uint32_t PCLK_CTL1_bit, FunctionalState NewState)

24.4.2 函数 USART_Configuration

表 24-6 函数 USART_Configuration

函数名	USART_Configuration
函数原型	void USART_Configuration (USART_SFRmap USARTx, USART_InitTypeDef usartInitStruct)
功能描述	配置 USART 外设初始化。
输入参数 1	USARTx:指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
输入参数 2	USARTInitStruct: USART 配置信息结构体指针。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.3 函数 USART_U7816R_Configuration

表 24-7 函数 USART_U7816R_Configuration

函数名	USART_U7816R_Configuration
函数原型	void USART_U7816R_Configuration (USART_SFRmap USARTx, U7816R_InitTypeDef usartInitStruct)
功能描述	7816 外设配置。
输入参数 1	USARTx:指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
输入参数 2	USARTInitStruct:USART_U7816 配置信息
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.4 函数 USART_Struct_Init

表 24-8 函数 USART_Struct_Init

函数名	USART_Struct_Init
函数原型	void USART_Struct_Init(USART_InitTypeDef usartInitStruct)
功能描述	初始化 USART 配置信息结构体。
输入参数 1	usartInitStruct: 指向待初始化的结构体指针。
输入参数 2	无
返回值	无
被调用函数	无

24.4.5 函数 USART_U7816R_Struct_Init

表 24-9 函数 USART_U7816R_Struct_Init

函数名	USART_U7816R_Struct_Init
函数原型	void USART_U7816R_Struct_Init(U7816R_InitTypeDef usartInitStruct)
功能描述	初始化 USART_7816 配置信息结构体。
输入参数 1	usartInitStruct: 指向待初始化的结构体指针。
输入参数 2	无
返回值	无
被调用函数	无

24.4.6 函数 USART_Cmd

表 24-10 函数 USART_Cmd

函数名	USART_Cmd
-----	-----------

函数原型	void USART_Cmd (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	配置 USART 使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
输入参数 2	NewState: USART 使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.7 函数 USART_BaudRate_Clock_Config

表 24-11 函数 USART_BaudRate_Clock_Config

函数名	USART_BaudRate_Clock_Config
函数原型	void USART_BaudRate_Clock_Config(USART_SFRmap USARTx, uint32_t CLK)
功能描述	配置 USART 波特率发生器时钟选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
输入参数 2	CLK: 时钟选择, 取值范围为: USART_CLK_SCLK :主时钟 USART_CLK_HFCLK :高频外设时钟 USART_CLK_LFCLK :低频外设时钟
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.8 函数 USART_HalfDuplex_ClockPolarity_Config

表 24-12 函数 USART_HalfDuplex_ClockPolarity_Config

函数名	USART_HalfDuplex_ClockPolarity_Config
函数原型	void USART_HalfDuplex_ClockPolarity_Config(USART_SFRmap USARTx, uint32_t NewState)
功能描述	设置 USART 半双工同步时钟极性选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
输入参数 2	NewState: USART 半双工同步时钟极性, 取值为: USART_CPOL_HIGH:数据在时钟下降沿同步, USART_CPOL_LOW :数据在时钟上升沿同步。
返回值	无
被调用函数	无

24.4.9 函数 USART_Transmit_Order_Config

表 24-13 函数 USART_Transmit_Order_Config

函数名	USART_Transmit_Order_Config
函数原型	void USART_Transmit_Order_Config (USART_SFRmap USARTx, uint32_t NewState)
功能描述	设置 USART 发送次序选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR。
输入参数 2	NewState: USART 发送次序, 取值为: USART_TRANSMIT_MSB:先发送 MSB, USART_TRANSMIT_LSB:先发送 LSB。
返回值	无
被调用函数	无

24.4.10 函数 USART_Receive_Order_Config

表 24-14 函数 USART_Receive_Order_Config

函数名	USART_Receive_Order_Config
函数原型	void USART_Receive_Order_Config (USART_SFRmap USARTx, uint32_t NewState)
功能描述	设置 USART 接收次序选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 接收次序, 取值为: USART_RECEIVE_MSB:先接收 MSB, USART_RECEIVE_LSB:先接收 LSB。
返回值	无
被调用函数	无

24.4.11 函数 USART_Infrare_Detector_Voltage_Config

表 24-15 函数 USART_Infrare_Detector_Voltage_Config

函数名	USART_Infrare_Detector_Voltage_Config
函数原型	void USART_Infrare_Detector_Voltage_Config(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置红外调制高电平输出极性。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: 红外调制高电平输出极性, 取值为 TRUE: 发送和接收反向电平,

	FALSE: 发送和接收正向电平。
返回值	无
被调用函数	无

24. 4. 12 函数 USART_WeakUP_Enable

表 24- 16 函数 USART_WeakUP_Enable

函数名	USART_WeakUP_Enable
函数原型	void USART_WeakUP_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 唤醒使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 唤醒使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 13 函数 USART_Clock_Source_Config

表 24- 17 函数 USART_Clock_Source_Config

函数名	USART_Clock_Source_Config
函数原型	void USART_Clock_Source_Config(USART_SFRmap USARTx, uint32_t ClockSource)
功能描述	设置 USART 时钟源选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	ClockSource: USART 时钟源选择, 取值为 TRUE 或 FALSE。 USART_MASTER_CLOCKSOURCE_INTER: 主模式(由 BRG 内部产生时钟), USART_SLAVE_CLOCKSOURCE_EXTER: 从模式(时钟源来自外部)。
返回值	无
被调用函数	无

24. 4. 14 函数 USART_Address_Detection_Enable

表 24- 18 函数 USART_Address_Detection_Enable

函数名	USART_Address_Detection_Enable
函数原型	void USART_Address_Detection_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 地址检测使能位。

输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 地址检测使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.15 函数 USART_Auto_BaudRate_Detection_Enable

表 24-19 函数 USART_Auto_BaudRate_Detection_Enable

函数名	USART_Auto_BaudRate_Detection_Enable
函数原型	void USART_Auto_BaudRate_Detection_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 自动波特率检测使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	无
返回值	无
被调用函数	无

24.4.16 函数 USART_Get_Auto_BaudRate_Detection_Flag

表 24-20 函数 USART_Get_Auto_BaudRate_Detection_Flag

函数名	USART_Get_Auto_BaudRate_Detection_Flag
函数原型	FlagStatus USART_Get_Auto_BaudRate_Detection_Flag(USART_SFRmap USARTx)
功能描述	读取 USART 自动波特率检测使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 自动波特率检测使能状态, 取值为 TRUE 或 FALSE。
返回值	1: 使能自动波特率模式(完成自动波特率后清零) 0: 禁止自动波特率模式
被调用函数	无

24.4.17 函数 USART_Send_Blank_Enable

表 24-21 函数 USART_Send_Blank_Enable

函数名	USART_Send_Blank_Enable
函数原型	void USART_Send_Blank_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 发送间隔字符使能。

输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 发送间隔字符使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 18 函数 USART_SYN_Choice_Config

表 24-22 函数 USART_SYN_Choice_Config

函数名	USART_SYN_Choice_Config
函数原型	void USART_SYN_Choice_Config(USART_SFRmap USARTx, uint32_t NewMode)
功能描述	设置 USART 串行通信模式选择位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewMode: USART 模式选择, 取值为: USART_MODE_HALFDUPLEXSYN:半双工同步模式, USART_MODE_FULLDUPLEXASY:全双工异步模式。
返回值	无
被调用函数	无

24. 4. 19 函数 USART_Transmit_Data_Enable

表 24-23 函数 USART_Transmit_Data_Enable

函数名	USART_Transmit_Data_Enable
函数原型	void USART_Transmit_Data_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 发送使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 发送使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 20 函数 USART_Receive_Data_Enable

表 24-24 函数 USART_Receive_Data_Enable

函数名	USART_Receive_Data_Enable
函数原型	void USART_Receive_Data_Enable (USART_SFRmap USARTx, FunctionalState NewState)

功能描述	设置 USART 接收使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 接收使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 21 函数 USART_STOP_Word_Config

表 24-25 函数 USART_STOP_Word_Config

函数名	USART_STOP_Word_Config
函数原型	void USART_STOP_Word_Config(USART_SFRmap USARTx, uint32_t NewLength)
功能描述	设置 USART 停止位长度选择位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewLength: 取值为 USART_STOPBITS_2 或 USART_STOPBITS_1。 USART_STOPBITS_2 :停止位长度为 2bits, USART_STOPBITS_1 :停止位长度为 1bit。
返回值	无
被调用函数	无

24. 4. 22 函数 USART_Transmit_9Word_Select_Config

表 24-26 函数 USART_Transmit_9Word_Select_Config

函数名	USART_Transmit_9Word_Select_Config
函数原型	void USART_Transmit_9Word_Select_Config (USART_SFRmap USARTx, uint32_t NewState)
功能描述	设置 USART 发送数据第 9 位数据选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: 取值为 USART_BIT9_RS485 或 USART_BIT9_PAR。 USART_BIT9_RS485 :RS-485 模式, USART_BIT9_PAR:选择奇偶校验。
返回值	无
被调用函数	无

24.4.23 函数 USART_Parity_Select_Config

表 24-27 函数 USART_Parity_Select_Config

函数名	USART_Parity_Select_Config
函数原型	void USART_Parity_Select_Config(USART_SFRmap USARTx, uint32_t NewState)
功能描述	设置 USART 奇偶校验选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: 取值为 USART_PARITY_EVEN 或 USART_PARITY_ODD。 USART_PARITY_EVEN:偶校验, USART_PARITY_ODD:奇校验。
返回值	无
被调用函数	无

24.4.24 函数 USART_9Data_Enable

表 24-28 函数 USART_9Data_Enable

函数名	USART_9Data_Enable
函数原型	void USART_9Data_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 第 9 位数据使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 第 9 位数据使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.25 函数 USART_CTS_Enable

表 24-29 函数 USART_CTS_Enable

函数名	USART_CTS_Enable
函数原型	void USART_CTS_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART CTS 使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART CTS 使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.26 函数 USART_RTS_Enable

表 24-30 函数 USART_RTS_Enable

函数名	USART_RTS_Enable
函数原型	void USART_RTS_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART RTS 使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART RTS 使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.27 函数 USART_Infrare_Detector_Enable

表 24-31 函数 USART_Infrare_Detector_Enable

函数名	USART_Infrare_Detector_Enable
函数原型	void USART_Infrare_Detector_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	NewState: USART 红外调制使能状态, 取值为 TRUE 或 FALSE。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 红外调制使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.28 函数 USART_RESHD_Enable

表 24-32 函数 USART_RESHD_Enable

函数名	USART_RESHD_Enable
函数原型	void USART_RESHD_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART RESHD 位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 单线模式使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.29 函数 USART_Singlet_Line_Mode_Enable

表 24-33 函数 USART_Singlet_Line_Mode_Enable

函数名	USART_Singlet_Line_Mode_Enable
函数原型	void USART_Singlet_Line_Mode_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 单线模式使能位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 单线模式使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.30 函数 USART_BaudRate_Integer_Config

表 24-34 函数 USART_BaudRate_Integer_Config

函数名	USART_BaudRate_Integer_Config
函数原型	void USART_BaudRate_Integer_Config(USART_SFRmap USARTx, uint16_t DIV)
功能描述	配置 USART USARTDIV 的整数部分。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	DIV: 取值范围为 0~65535。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.31 函数 USART_BaudRate_Decimal1_Config

表 24-35 函数 USART_BaudRate_Decimal1_Config

函数名	USART_BaudRate_Decimal1_Config
函数原型	void USART_BaudRate_Decimal1_Config(USART_SFRmap USARTx, uint32_t DIV)
功能描述	配置 USART 小数波特率分子。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	DIV: 取值范围为 0~15。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.32 函数 USART_BaudRate_Decimal2_Config

表 24-36 函数 USART_BaudRate_Decimal2_Config

函数名	USART_BaudRate_Decimal2_Config
函数原型	void USART_BaudRate_Decimal2_Config(USART_SFRmap USARTx, uint32_t DIV)
功能描述	配置 USART 小数波特率分母。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	DIV: 取值范围为 0~15。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.33 函数 USART_SendData

表 24-37 函数 USART_SendData

函数名	USART_SendData
函数原型	void USART_SendData(USART_SFRmap USARTx, uint8_t Data)
功能描述	USART 发送数据。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	Data: 写入数据寄存器的值, 取值为 0~255。
返回值	无
被调用函数	无

24.4.34 函数 USART_TransmitData

表 24-38 函数 USART_TransmitData

函数名	USART_TransmitData
函数原型	void USART_TransmitData(USART_SFRmap USARTx, uint8_t Data)
功能描述	USART 发送字节,并等待发送器和缓冲器空。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	Data: 写入数据寄存器的值, 取值为 0~255。
返回值	无
被调用函数	无

24.4.35 函数 USART_ReceiveData

表 24-39 函数 USART_ReceiveData

函数名	USART_ReceiveData
函数原型	uint32_t USART_ReceiveData(USART_SFRmap USARTx)
功能描述	USART 接收数据。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	无
返回值	32 位类型数据的 8 位数据。
被调用函数	无

24.4.36 函数 USART_Address_Match_Config

表 24-40 函数 USART_Address_Match_Config

函数名	USART_Address_Match_Config
函数原型	void USART_Address_Match_Config(USART_SFRmap USARTx, uint8_t DIV)
功能描述	配置 USARTx 地址匹配功能设置位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	address: USARTx 地址匹配功能, 取值范围为 0~255。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.37 函数 USART_Send_Idle_Frame_Enable

表 24-41 函数 USART_Send_Idle_Frame_Enable

函数名	USART_Send_Idle_Frame_Enable
函数原型	void USART_Send_Idle_Frame_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	配置 USART 全双工模式时发送空闲帧使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState:TRUE: 使能发送器发送空闲帧 FALSE: 禁止发送器发送空闲帧。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24. 4. 38 函数 USART_Receive_Idle_Frame_Config

表 24-42 函数 USART_Receive_Idle_Frame_Config

函数名	USART_Receive_Idle_Frame_Config
函数原型	void USART_Receive_Idle_Frame_Config(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	配置 USART 接收空闲帧中断标志产生模式。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: TRUE: 空闲帧中断标志和 RDRIF 无关 FALSE: 空闲帧中断标志和 RDRIF 有关。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24. 4. 39 函数 USART_7816_Cmd

表 24-43 函数 USART_7816_Cmd

函数名	USART_7816_Cmd
函数原型	void USART_7816_Cmd(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 7816 模式使能控制。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 7816 模式使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 40 函数 USART_7816_CLKOUT_Enable

表 24-44 函数 USART_7816_CLKOUT_Enable

函数名	USART_7816_CLKOUT_Enable
函数原型	void USART_7816_CLKOUT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 7816 时钟输出使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 7816 时钟输出使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.41 函数 USART_7816_Error_Signal_Config

表 24-45 函数 USART_7816_Error_Signal_Config

函数名	USART_7816_Error_Signal_Config
函数原型	void USART_7816_Error_Signal_Config (USART_SFRmap USARTx, uint32_t ERRORSIGNAL)
功能描述	配置 USART 7816 error signal 宽度选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	signal: 宽度选择, 取值范围为: USART_U7816R_ERRORSIGNAL_2ETU : 2etu USART_U7816R_ERRORSIGNAL_1P5ETU : 1.5etu USART_U7816R_ERRORSIGNAL_1ETU : 1etu
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.42 函数 USART_Passageway_Select_Config

表 24-46 函数 USART_Passageway_Select_Config

函数名	USART_Passageway_Select_Config
函数原型	void USART_Passageway_Select_Config(USART_SFRmap USARTx, uint32_t NewState)
功能描述	设置 USART 数据通道选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: 取值为 USART_U7816R_PASSAGEWAY_TX1 或 USART_U7816R_PASSAGEWAY_TX0。 USART_U7816R_PASSAGEWAY_TX1 :选择通道 1(TX1), USART_U7816R_PASSAGEWAY_TX0 :选择通道 0(TX0)
返回值	无
被调用函数	无

24.4.43 函数 USART_BGT_Config

表 24-47 函数 USART_BGT_Config

函数名	USART_BGT_Config
函数原型	void USART_BGT_Config(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 控制接收到发送之间是否插入 BGT。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为

	USART0_SFR~USART8_SFR
输入参数 2	NewState: 取值为 TRUE 或 FALSE。 TRUE:插入 BGT, 宽度为 22etu, FALSE:不插入 BGT
返回值	无
被调用函数	无

24. 4. 44 函数 USART_Transmit_Repeat_Enable

表 24-48 函数 USART_Transmit_Repeat_Enable

函数名	USART_Transmit_Repeat_Enable
函数原型	void USART_Transmit_Repeat_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 重发使能控制。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: 取值为 TRUE 或 FALSE。 TRUE:收到校验错误信号后重发, 达到最大次数后进入中断, FALSE:收到校验错误信号后直接进入中断
返回值	无
被调用函数	无

24. 4. 45 函数 USART_Receive_Repeat_Enable

表 24-49 函数 USART_Receive_Repeat_Enable

函数名	USART_Receive_Repeat_Enable
函数原型	void USART_Receive_Repeat_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 重收使能控制。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: 取值为 TRUE 或 FALSE。 TRUE:奇偶校验错误后重收数据, 达到最大次数后进入中断 FALSE:奇偶校验错误后直接进入中断
返回值	无
被调用函数	无

24. 4. 46 函数 USART_Transmit_Repeat_Times_Config

表 24-50 函数 USART_Transmit_Repeat_Times_Config

函数名	USART_Transmit_Repeat_Times_Config
函数原型	void USART_Transmit_Repeat_Times_Config(USART_SFRmap USARTx, uint32_t SELECT)
功能描述	配置 USART 奇偶校验出错时重发送的最大次数。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	SELECT: 最大次数选择, 取值范围为: USART_U7816R_TRANSMIT_REPEAT_0S: 最大发送收次数为 0 USART_U7816R_TRANSMIT_REPEAT_1S: 最大发送收次数为 1 USART_U7816R_TRANSMIT_REPEAT_2S: 最大发送收次数为 2 USART_U7816R_TRANSMIT_REPEAT_3S: 最大发送收次数为 3
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24. 4. 47 函数 USART_Receive_Repeat_Times_Config

表 24-51 函数 USART_Receive_Repeat_Times_Config

函数名	USART_Receive_Repeat_Times_Config
函数原型	void USART_Receive_Repeat_Times_Config(USART_SFRmap USARTx, uint32_t SELECT)
功能描述	配置 USART 奇偶校验出错时重接收的最大次数。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	SELECT: 最大次数选择, 取值范围为: USART_U7816R_RECEIVE_REPEAT_0S: 最大发送收次数为 0 USART_U7816R_RECEIVE_REPEAT_1S: 最大发送收次数为 1 USART_U7816R_RECEIVE_REPEAT_2S: 最大发送收次数为 2 USART_U7816R_RECEIVE_REPEAT_3S: 最大发送收次数为 3
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24. 4. 48 函数 USART_7816_CLKDIV_Config

表 24-52 函数 USART_7816_CLKDIV_Config

函数名	USART_7816_CLKDIV_Config
函数原型	void USART_7816_CLKDIV_Config (USART_SFRmap USARTx, uint8_t

	DIV)
功能描述	配置 USART 7816 工作时钟和引脚输出时钟控制。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	CLKDIV: 取值范围为 0~255。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24. 4. 49 函数 USART_7816_EGT_Config

表 24-53 函数 USART_7816_EGT_Config

函数名	USART_7816_EGT_Config
函数原型	void USART_7816_EGT_Config(USART_SFRmap USARTx, uint8_t EGT)
功能描述	配置 USART 7816 发送时插入的 EGT(extra guard time) (单位 etu) 。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	EGT: 取值范围为 0~255。
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24. 4. 50 函数 USART_7816_Resend_Mode_Select

表 24-54 函数 USART_7816_Resend_Mode_Select

函数名	USART_7816_Resend_Mode_Select
函数原型	void USART_7816_Resend_Mode_Select(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	配置 USART 7816 重发模式选择。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: FALSE: 只要接收到错误帧就启动重发 TRUE: 重发次数到设置次数后停止重发
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal)

24.4.51 函数 USART_Receive_Overflow_INT_Enable

表 24-55 函数 USART_Receive_Overflow_INT_Enable

函数名	USART_Receive_Overflow_INT_Enable
函数原型	void USART_Receive_Overflow_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 接收溢出中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 接收溢出中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.52 函数 USART_Parity_ERROR_INT_Enable

表 24-56 函数 USART_Parity_ERROR_INT_Enable

函数名	USART_Parity_ERROR_INT_Enable
函数原型	void USART_Parity_ERROR_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 奇偶校验错误中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 奇偶校验错误中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.53 函数 USART_Frame_ERROE_INT_Enable

表 24-57 函数 USART_Frame_ERROE_INT_Enable

函数名	USART_Frame_ERROE_INT_Enable
函数原型	void USART_Frame_ERROE_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 帧错误中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 帧错误中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 54 函数 USART_Blank_INT_Enable

表 24- 58 函数 USART_Blank_INT_Enable

函数名	USART_Blank_INT_Enable
函数原型	void USART_Blank_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 间隔符位中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 间隔符位中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 55 函数 USART_Auto_BaudRate_TimeOver_INT_Enable

表 24- 59 函数 USART_Auto_BaudRate_TimeOver_INT_Enable

函数名	USART_Auto_BaudRate_TimeOver_INT_Enable
函数原型	void USART_Auto_BaudRate_TimeOver_INT_Enable (USART_SFRmap USARTx,FunctionalState NewState)
功能描述	设置 USART 自动波特率超时中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 自动波特率超时中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 56 函数 USART_WeakUP_INT_Enable

表 24- 60 函数 USART_WeakUP_INT_Enable

函数名	USART_WeakUP_INT_Enable
函数原型	void USART_WeakUP_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 自动唤醒中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 自动唤醒中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.57 函数 USART_Transmit_ERROR_INT_Enable

表 24-61 函数 USART_Transmit_ERROR_INT_Enable

函数名	USART_Transmit_ERROR_INT_Enable
函数原型	void USART_Transmit_ERROR_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART7816 发送错误中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART7816 发送错误中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.58 函数 USART_Receive_ERROR_INT_Enable

表 24-62 函数 USART_Receive_ERROR_INT_Enable

函数名	USART_Receive_ERROR_INT_Enable
函数原型	void USART_Receive_ERROR_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART7816 接收错误中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART7816 接收错误中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.59 函数 USART_CTS_INT_Enable

表 24-63 函数 USART_CTS_INT_Enable

函数名	USART_CTS_INT_Enable
函数原型	void USART_CTS_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART CTS 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART CTS 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.60 函数 USART_RDR_INT_Enable

表 24-64 函数 USART_RDR_INT_Enable

函数名	USART_RDR_INT_Enable
函数原型	void USART_RDR_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART RDR 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART RDR 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.61 函数 USART_TFE_INT_Enable

表 24-65 函数 USART_TFE_INT_Enable

函数名	USART_TFE_INT_Enable
函数原型	void USART_TFE_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART TFE 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART TFE 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24.4.62 函数 USART_TXE_INT_Enable

表 24-66 函数 USART_TXE_INT_Enable

函数名	USART_TXE_INT_Enable
函数原型	void USART_TXE_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART TXE 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART TXE 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 63 函数 USART_Receive_DMA_INT_Enable

表 24-67 函数 USART_Receive_DMA_INT_Enable

函数名	USART_Receive_DMA_INT_Enable
函数原型	void USART_Receive_DMA_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 接收数据 DMA 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 接收数据 DMA 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 64 函数 USART_Transmit_DMA_INT_Enable

表 24-68 函数 USART_Transmit_DMA_INT_Enable

函数名	USART_Transmit_DMA_INT_Enable
函数原型	void USART_Transmit_DMA_INT_Enable (USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART 发送数据 DMA 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 发送数据 DMA 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 65 函数 USART_IDLE_INT_Enable

表 24-69 函数 USART_IDLE_INT_Enable

函数名	USART_IDLE_INT_Enable
函数原型	void USART_IDLE_INT_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART IDLEIF 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 66 函数 USART_UADM_INT_Enable

表 24- 70 函数 USART_UADM_INT_Enable

函数名	USART_UADM_INT_Enable
函数原型	void USART_UADM_INT_Enable(USART_SFRmap USARTx, FunctionalState NewState)
功能描述	设置 USART UADMIF 中断使能。
输入参数 1	USARTx: 指向 USART 内存结构的指针，取值为 USART0_SFR~USART8_SFR
输入参数 2	NewState: USART 中断使能状态，取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

24. 4. 67 函数 USART_Get_Receive_Overflow_Flag

表 24- 71 函数 USART_Get_Receive_Overflow_Flag

函数名	USART_Get_Receive_Overflow_Flag
函数原型	FlagStatus USART_Get_Receive_Overflow_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 接收溢出中断标志状态 。
输入参数 1	USARTx: 指向 USART 内存结构的指针，取值为 USART0_SFR~USART8_SFR
返回值	1: USART 接收溢出； 0: USART 没有接收溢出。
被调用函数	无

24. 4. 68 函数 USART_Get_Parity_ERROR_Flag

表 24- 72 函数 USART_Get_Parity_ERROR_Flag

函数名	USART_Get_Parity_ERROR_Flag
函数原型	FlagStatus USART_Get_Parity_ERROR_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 奇偶校验错误中断标志状态 。
输入参数 1	USARTx: 指向 USART 内存结构的指针，取值为 USART0_SFR~USART8_SFR
返回值	1: USART 奇偶校验错误； 0: USART 没有接收溢出。
被调用函数	无

24.4.69 函数 USART_Get_Frame_ERROR_Flag

表 24-73 函数 USART_Get_Frame_ERROR_Flag

函数名	USART_Get_Frame_ERROR_Flag
函数原型	FlagStatus USART_Get_Frame_ERROR_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 帧错误中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 接收数据时发生帧错误; 0: USART 接收数据时未发生帧错误。
被调用函数	无

24.4.70 函数 USART_Get_Blank_Flag

表 24-74 函数 USART_Get_Blank_Flag

函数名	USART_Get_Blank_Flag
函数原型	FlagStatus USART_Get_Blank_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 间隔符中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 接收到间隔字符; 0: USART 未接受到间隔字符。
被调用函数	无

24.4.71 函数 USART_Get_Auto_Baudrate_TimeOver_Flag

表 24-75 函数 USART_Get_Auto_Baudrate_TimeOver_Flag

函数名	USART_Get_Auto_Baudrate_TimeOver_Flag
函数原型	FlagStatus USART_Get_Auto_Baudrate_TimeOver_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 自动波特率超时中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 自动波特率超时; 0: USART 自动波特率未超时。
被调用函数	无

24.4.72 函数 USART_Get_WeakUP_Flag

表 24-76 函数 USART_Get_WeakUP_Flag

函数名	USART_Get_WeakUP_Flag
函数原型	FlagStatus USART_Get_WeakUP_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 自动唤醒中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 发生了自动唤醒事件; 0: USART 未发生自动唤醒事件。
被调用函数	无

24.4.73 函数 USART_Get_7816Transmit_ERROR_Flag

表 24-77 函数 USART_Get_7816Transmit_ERROR_Flag

函数名	USART_Get_7816Transmit_ERROR_Flag
函数原型	FlagStatus USART_Get_7816Transmit_ERROR_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 7816 发送错误中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 产生了 7816 发送错误中断; 0: USART 未产生 7816 发送错误中断。
被调用函数	无

24.4.74 函数 USART_Get_7816Receive_ERROR_Flag

表 24-78 函数 USART_Get_7816Receive_ERROR_Flag

函数名	USART_Get_7816Receive_ERROR_Flag
函数原型	FlagStatus USART_Get_7816Receive_ERROR_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 7816 接收错误中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 产生了 7816 接收错误中断; 0: USART 未产生 7816 接收错误中断。
被调用函数	无

24. 4. 75 函数 USART_Get_CTS_Flag

表 24- 79 函数 USART_Get_CTS_Flag

函数名	USART_Get_CTS_Flag
函数原型	FlagStatus USART_Get_CTS_Flag (USART_SFRmap USARTx)
功能描述	获取 USART CTS 中断标志状态 。
输入参数 1	USARTx: 指向 USART 内存结构的指针，取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24. 4. 76 函数 USART_Get_Receive_BUFR_Ready_Flag

表 24- 80 函数 USART_Get_Receive_BUFR_Ready_Flag

函数名	USART_Get_Receive_BUFR_Ready_Flag
函数原型	FlagStatus USART_Get_Receive_BUFR_Ready_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 数据就绪中断标志状态 。
输入参数 1	USARTx: 指向 USART 内存结构的指针，取值为 USART0_SFR~USART8_SFR
输入参数 2	1: USART 接收 BUF 中有数据可读； 0: USART 接收 BUF 中无数据可读。
返回值	无
被调用函数	无

24. 4. 77 函数 USART_Get_WUEN_Flag

表 24- 81 函数 USART_Get_WUEN_Flag

函数名	USART_Get_WUEN_Flag
函数原型	FlagStatus USART_Get_WUEN_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 唤醒使能位状态 。
输入参数 1	USARTx: 指向 USART 内存结构的指针，取值为 USART0_SFR~USART8_SFR
返回值	1: USART 正在等待下降沿； 0: USART 接收器正常工作。
被调用函数	无

24.4.78 函数 USART_Get_Transmit_BUFR_Empty_Flag

表 24-82 函数 USART_Get_Transmit_BUFR_Empty_Flag

函数名	USART_Get_Transmit_BUFR_Empty_Flag
函数原型	FlagStatus USART_Get_Transmit_BUFR_Empty_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 发送 BUF 为空中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 发送 BUF 为空; 0: USART 发送 BUF 不为空。
被调用函数	无

24.4.79 函数 USART_Get_Transmitter_Empty_Flag

表 24-83 函数 USART_Get_Transmitter_Empty_Flag

函数名	USART_Get_Transmitter_Empty_Flag
函数原型	FlagStatus USART_Get_Transmitter_Empty_Flag (USART_SFRmap USARTx)
功能描述	获取 USART 发射器为空中断标志状态。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: USART 发射器为空; 0: USART 发射器不为空。
被调用函数	无

24.4.80 函数 USART_Get_Receive_Frame_Idel_Flag

表 24-84 函数 USART_Get_Receive_Frame_Idel_Flag

函数名	USART_Get_Receive_Frame_Idel_Flag
函数原型	FlagStatus USART_Get_Receive_Frame_Idel_Flag(USART_SFRmap USARTx)
功能描述	获取 USART 接收空闲帧中断标志。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	1: 接收器接收到空闲帧 0: 接收器没有接收到空闲帧。
被调用函数	无

24.4.81 函数 USART_Clear_Receive_Overflow_INT_Flag

表 24-85 函数 USART_Clear_Receive_Overflow_INT_Flag

函数名	USART_Clear_Receive_Overflow_INT_Flag
函数原型	void USART_Clear_Receive_Overflow_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 接收溢出中断标志。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.82 函数 USART_Clear_Parity_ERROR_INT_Flag

表 24-86 函数 USART_Clear_Parity_ERROR_INT_Flag

函数名	USART_Clear_Parity_ERROR_INT_Flag
函数原型	void USART_Clear_Parity_ERROR_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 奇偶校验错误位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.83 函数 USART_Clear_Frame_ERROR_INT_Flag

表 24-87 函数 USART_Clear_Frame_ERROR_INT_Flag

函数名	USART_Clear_Frame_ERROR_INT_Flag
函数原型	void USART_Clear_Frame_ERROR_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 帧错误位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.84 函数 USART_Clear_Blank_INT_Flag

表 24-88 函数 USART_Clear_Blank_INT_Flag

函数名	USART_Clear_Blank_INT_Flag
函数原型	void USART_Clear_Blank_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 间隔符位。

输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24. 4. 85 函数 USART_Clear_Auto_BaudRate_TimeOver_INT_Flag

表 24- 89 函数 USART_Clear_Auto_BaudRate_TimeOver_INT_Flag

函数名	USART_Clear_Auto_BaudRate_TimeOver_INT_Flag
函数原型	void USART_Clear_Auto_BaudRate_TimeOver_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 自动波特率超时位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24. 4. 86 函数 USART_Clear_WeakUP_INT_Flag

表 24- 90 函数 USART_Clear_WeakUP_INT_Flag

函数名	USART_Clear_WeakUP_INT_Flag
函数原型	void USART_Clear_WeakUP_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 自动唤醒中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24. 4. 87 函数 USART_Clear_Transmit_ERROR_INT_Flag

表 24- 91 函数 USART_Clear_Transmit_ERROR_INT_Flag

函数名	USART_Clear_Transmit_ERROR_INT_Flag
函数原型	void USART_Clear_Transmit_ERROR_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 发送错误中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.88 函数 USART_Clear_Receive_ERROR_INT_Flag

表 24-92 函数 USART_Clear_Receive_ERROR_INT_Flag

函数名	USART_Clear_Receive_ERROR_INT_Flag
函数原型	void USART_Clear_Receive_ERROR_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 接收错误中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.89 函数 USART_Clear_CTS_INT_Flag

表 24-93 函数 USART_Clear_CTS_INT_Flag

函数名	USART_Clear_CTS_INT_Flag
函数原型	void USART_Clear_CTS_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART CTS 中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.90 函数 USART_Clear_UADM_INT_Flag

表 24-94 函数 USART_Clear_UADM_INT_Flag

函数名	USART_Clear_UADM_INT_Flag
函数原型	void USART_Clear_UADM_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART UADM 中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24.4.91 函数 USART_Clear_IDLE_INT_Flag

表 24-95 函数 USART_Clear_IDLE_INT_Flag

函数名	USART_Clear_IDLE_INT_Flag
函数原型	void USART_Clear_IDLE_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART IDLE 中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为

	USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24. 4. 92 函数 USART_Clear_Receive_BUFR_INT_Flag

表 24-96 函数 USART_Clear_Receive_BUFR_INT_Flag

函数名	USART_Clear_Receive_BUFR_INT_Flag
函数原型	void USART_Clear_Receive_BUFR_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 接收 BUF 中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

24. 4. 93 函数 USART_Clear_Transmit_BUFR_INT_Flag

表 24-97 函数 USART_Clear_Transmit_BUFR_INT_Flag

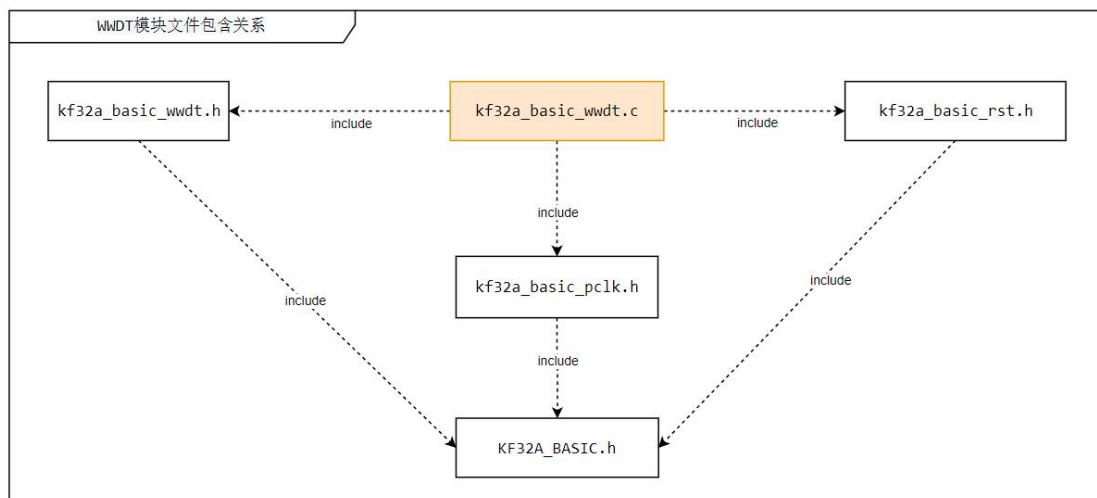
函数名	USART_Clear_Transmit_BUFR_INT_Flag
函数原型	void USART_Clear_Transmit_BUFR_INT_Flag (USART_SFRmap USARTx)
功能描述	清零 USART 发送 BUF 中断位。
输入参数 1	USARTx: 指向 USART 内存结构的指针, 取值为 USART0_SFR~USART8_SFR
返回值	无
被调用函数	无

25 窗口看门狗（WWDT）

窗口看门狗通常被用来监测由外部干扰或不可预见的逻辑条件造成的应用程序背离正常的运行序列而产生的软件故障。WWDT 最适合那些要求看门狗在精确计时窗口起作用的应用程序。通过可配置的时间窗口来检测应用程序非正常的过迟或过早的操作。

25.1 文件引用关系

图 25-1 WWDT 模块文件包含关系



25.2 WWDT 寄存器结构

WWDT 寄存器结构，WWDT_SFRmap，定义于文件 KF32A_BASIC.h 中，如下：

```

typedef struct WWDT_MemMap {
    volatile uint32_t    CTL;
    volatile uint32_t    CNT;
} WWDT_SFRmap;
    
```

表 25-1 WWDT 寄存器结构说明

寄存器	描述
CTL	独立看门狗控制寄存器，偏移:0x00
CNT	独立看门狗计数寄存器，偏移:0x04

25.3 WWDT 宏定义

WWDT 寄存器入口地址、寄存器入口、位域的宏定义详见文件 KF32A_BASIC.h，WWDT 其他相关宏定义详见文件 kf32a_basic_wwdt.h 及函数参数描述。

25.4 WWDT 库函数

表 25-2 WWDT 固件库函数列表

序号	函数名	描述
1	WWDT_Reset	复位 WWDT 模块，使能外设时钟。
2	WWDT_Threshold_Config	设置窗口看门狗可操作区下限值。
3	WWDT_Prescaler_Config	设置窗口看门狗预分频，对 INTLF 进行分频。
4	WWDT_Enable	设置窗口看门狗使能。
5	WWDT_Counter_Config	配置窗口看门狗计数值。
6	WWDT_Get_Counter	获取窗口看门狗计数值。
7	WWDT_INT_Enable	设置窗口看门狗中断使能。
8	WWDT_Get_INT_Flag	获取窗口看门狗中断标志。
9	WWDT_Clear_INT_Flag	清零窗口看门狗中断标志。

25.4.1 函数 WWDT_Reset

表 25-3 函数 WWDT_Reset

函数名	WWDT_Reset
函数原型	void WWDT_Reset(void)
功能描述	复位 WWDT 模块，使能外设时钟。
输入参数 1	无
返回值	无
被调用函数 1	void RST_CTL2_Peripheral_Reset_Enable(uint32_t RST_CTL2_bit, FunctionalState NewState);
被调用函数 2	void PCLK_CTL2_Peripheral_Clock_Enable(uint32_t PCLK_CTL2_bit, FunctionalState NewState);

25.4.2 函数 WWDT_Threshold_Config

表 25-4 函数 WWDT_Threshold_Config

函数名	WWDT_Threshold_Config
函数原型	void WWDT_Threshold_Config(uint32_t Threshold)
功能描述	设置窗口看门狗可操作区下限值。
输入参数 1	Threshold: 窗口看门狗可操作区下限值，取值为 0~0x3F。
返回值	无
被调用函数	static inline uint32_t SFR_Config(uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

25.4.3 函数 **WWDT_Prescaler_Config**

表 25-5 函数 WWDT_Prescaler_Config

函数名	WWDT_Prescaler_Config
函数原型	void WWDT_Prescaler_Config (uint32_t Prescaler)
功能描述	设置窗口看门狗预分频，对 INTLF 进行分频。
输入参数 1	Prescaler: 窗口看门狗对 INTLF 的预分频值，取值为： WWDT_PRESCALER_1: 不分频 WWDT_PRESCALER_2: 2 分频 WWDT_PRESCALER_4: 4 分频 WWDT_PRESCALER_8: 8 分频 WWDT_PRESCALER_16: 16 分频 WWDT_PRESCALER_32: 32 分频 WWDT_PRESCALER_64: 64 分频 WWDT_PRESCALER_128: 128 分频 WWDT_PRESCALER_256: 256 分频 WWDT_PRESCALER_512: 512 分频 WWDT_PRESCALER_1024: 1024 分频 WWDT_PRESCALER_2048: 2048 分频 WWDT_PRESCALER_4096: 4096 分频 WWDT_PRESCALER_8192: 8192 分频 WWDT_PRESCALER_16384: 16384 分频 WWDT_PRESCALER_32768: 32768 分频 WWDT_PRESCALER_65536: 65536 分频 WWDT_PRESCALER_131072: 131072 分频 WWDT_PRESCALER_262144: 262144 分频
返回值	无
被调用函数	static inline uint32_t SFR_Config (uint32_t SfrMem, uint32_t SfrMask, uint32_t WriteVal);

25.4.4 函数 **WWDT_Enable**

表 25-6 函数 WWDT_Enable

函数名	WWDT_Enable
函数原型	void WWDT_Enable (FunctionalState NewState)
功能描述	设置窗口看门狗使能。
输入参数 1	NewState: 窗口看门狗使能状态，取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

25.4.5 函数 WWDT_Counter_Config

表 25-7 函数 WWDT_Counter_Config

函数名	WWDT_Counter_Config
函数原型	void WWDT_Counter_Config (uint32_t Counter)
功能描述	配置窗口看门狗计数值。
输入参数 1	Counter: 窗口看门狗计数值, 取值为 7 位有效数值。
返回值	无
被调用函数	无

25.4.6 函数 WWDT_Get_Counter

表 25-8 函数 WWDT_Get_Counter

函数名	WWDT_Get_Counter
函数原型	uint32_t WWDT_Get_Counter (void)
功能描述	获取窗口看门狗计数值。
输入参数 1	无
返回值	窗口看门狗计数值, 7 位有效数值。
被调用函数	无

25.4.7 函数 WWDT_INT_Enable

表 25-9 函数 WWDT_INT_Enable

函数名	WWDT_INT_Enable
函数原型	void WWDT_INT_Enable (FunctionalState NewState)
功能描述	设置窗口看门狗中断使能。
输入参数 1	NewState: 窗口看门狗中断使能状态, 取值为 TRUE 或 FALSE。
返回值	无
被调用函数	无

25.4.8 函数 WWDT_Get_INT_Flag

表 25-10 函数 WWDT_Get_INT_Flag

函数名	WWDT_Get_INT_Flag
函数原型	FlagStatus WWDT_Get_INT_Flag (void)
功能描述	获取窗口看门狗中断标志。
输入参数 1	无
返回值	1:产生了窗口看门狗中断; 0:未产生窗口看门狗中断。
被调用函数	无

25.4.9 函数 WWDT_Clear_INT_Flag

表 25-11 函数 WWDT_Clear_INT_Flag

函数名	WWDT_Clear_INT_Flag
函数原型	void WWDT_Clear_INT_Flag (void)
功能描述	清零窗口看门狗中断标志。
输入参数 1	无
返回值	无
被调用函数	无

附录 2 历史使用手册版本信息

版本号	更新说明	更新日期
V1.0	初版	2021.4.18