

NIIETCM4 PD
v0.8.0

Создано системой Doxygen 1.8.10

Пн 21 Мар 2016 11:42:52

Оглавление

1	Титульная страница	1
2	Информация о релизах	3
3	Алфавитный указатель групп	7
3.1	Группы	7
4	Алфавитный указатель структур данных	11
4.1	Структуры данных	11
5	Список файлов	13
5.1	Файлы	13
6	Группы	15
6.1	Константы	15
6.1.1	Подробное описание	15
6.2	Маски каналов для измерений	16
6.2.1	Подробное описание	16
6.2.2	Макросы	16
6.2.2.1	ADC_Channel_0	16
6.2.2.2	ADC_Channel_1	16
6.2.2.3	ADC_Channel_10	16
6.2.2.4	ADC_Channel_11	17
6.2.2.5	ADC_Channel_12	17
6.2.2.6	ADC_Channel_13	17
6.2.2.7	ADC_Channel_14	17
6.2.2.8	ADC_Channel_15	17
6.2.2.9	ADC_Channel_16	17
6.2.2.10	ADC_Channel_17	17
6.2.2.11	ADC_Channel_18	17
6.2.2.12	ADC_Channel_19	17
6.2.2.13	ADC_Channel_2	17
6.2.2.14	ADC_Channel_20	18

6.2.2.15	ADC_Channel_21	18
6.2.2.16	ADC_Channel_22	18
6.2.2.17	ADC_Channel_23	18
6.2.2.18	ADC_Channel_3	18
6.2.2.19	ADC_Channel_4	18
6.2.2.20	ADC_Channel_5	18
6.2.2.21	ADC_Channel_6	18
6.2.2.22	ADC_Channel_7	18
6.2.2.23	ADC_Channel_8	19
6.2.2.24	ADC_Channel_9	19
6.2.2.25	ADC_Channel_All	19
6.2.2.26	ADC_Channel_None	19
6.3	Маски выбора цифровых компараторов	20
6.3.1	Подробное описание	20
6.3.2	Макросы	20
6.3.2.1	ADC_DC_0	20
6.3.2.2	ADC_DC_1	20
6.3.2.3	ADC_DC_10	20
6.3.2.4	ADC_DC_11	21
6.3.2.5	ADC_DC_12	21
6.3.2.6	ADC_DC_13	21
6.3.2.7	ADC_DC_14	21
6.3.2.8	ADC_DC_15	21
6.3.2.9	ADC_DC_16	21
6.3.2.10	ADC_DC_17	21
6.3.2.11	ADC_DC_18	21
6.3.2.12	ADC_DC_19	21
6.3.2.13	ADC_DC_2	21
6.3.2.14	ADC_DC_20	22
6.3.2.15	ADC_DC_21	22
6.3.2.16	ADC_DC_22	22
6.3.2.17	ADC_DC_23	22
6.3.2.18	ADC_DC_3	22
6.3.2.19	ADC_DC_4	22
6.3.2.20	ADC_DC_5	22
6.3.2.21	ADC_DC_6	22
6.3.2.22	ADC_DC_7	22
6.3.2.23	ADC_DC_8	23
6.3.2.24	ADC_DC_9	23
6.3.2.25	ADC_DC_All	23

6.3.2.26	ADC_DC_None	23
6.4	Маски выбора секвенсоров	24
6.4.1	Подробное описание	24
6.4.2	Макросы	24
6.4.2.1	ADC_SEQ_0	24
6.4.2.2	ADC_SEQ_1	24
6.4.2.3	ADC_SEQ_2	24
6.4.2.4	ADC_SEQ_3	24
6.4.2.5	ADC_SEQ_4	24
6.4.2.6	ADC_SEQ_5	24
6.4.2.7	ADC_SEQ_6	25
6.4.2.8	ADC_SEQ_7	25
6.5	Типы	26
6.5.1	Подробное описание	28
6.5.2	Макросы	28
6.5.2.1	IS_ADC_AVERAGE	28
6.5.2.2	IS_ADC_DC_CHANNEL	28
6.5.2.3	IS_ADC_DC_CONDITION	29
6.5.2.4	IS_ADC_DC_MODE	29
6.5.2.5	IS_ADC_DC_MODULE	29
6.5.2.6	IS_ADC_MEASURE	30
6.5.2.7	IS_ADC_MODE	30
6.5.2.8	IS_ADC_MODULE	30
6.5.2.9	IS_ADC_RESOLUTION	30
6.5.2.10	IS_ADC_SEQ_FIFO_LEVEL	31
6.5.2.11	IS_ADC_SEQ_MODULE	31
6.5.2.12	IS_ADC_SEQ_START_EVENT	31
6.5.3	Перечисления	32
6.5.3.1	ADC_Average_TypeDef	32
6.5.3.2	ADC_DC_Channel_TypeDef	32
6.5.3.3	ADC_DC_Condition_TypeDef	33
6.5.3.4	ADC_DC_Mode_TypeDef	33
6.5.3.5	ADC_DC_Module_TypeDef	33
6.5.3.6	ADC_Measure_TypeDef	34
6.5.3.7	ADC_Mode_TypeDef	34
6.5.3.8	ADC_Module_TypeDef	35
6.5.3.9	ADC_Resolution_TypeDef	35
6.5.3.10	ADC_SEQ_FIFOLevel_TypeDef	35
6.5.3.11	ADC_SEQ_Module_TypeDef	36
6.5.3.12	ADC_SEQ_StartEvent_TypeDef	36

6.6	Функции	37
6.6.1	Подробное описание	37
6.6.2	Функции	37
6.6.2.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	37
6.6.2.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	38
6.6.2.3	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_Module)	38
6.6.2.4	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	38
6.6.2.5	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC_Module)	39
6.6.2.6	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	39
6.6.2.7	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	39
6.6.2.8	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	40
6.6.2.9	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	40
6.6.2.10	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	40
6.6.2.11	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	41
6.6.2.12	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	41
6.6.2.13	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	41
6.6.2.14	ADC_SEQ_SWReq()	41
6.7	Инициализация	43
6.7.1	Подробное описание	43
6.8	Модули АЦП	44
6.8.1	Подробное описание	44
6.8.2	Функции	44
6.8.2.1	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	44
6.8.2.2	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *ADC_InitStruct)	44
6.8.2.3	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	44
6.9	Цифровые компараторы	46
6.9.1	Подробное описание	46
6.9.2	Функции	46
6.9.2.1	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module)	46
6.9.2.2	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	46

6.9.2.3	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	46
6.10	Секвенсоры	48
6.10.1	Подробное описание	48
6.10.2	Функции	48
6.10.2.1	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	48
6.10.2.2	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	48
6.10.2.3	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	49
6.11	Конфигурация секвенсоров для DMA	51
6.11.1	Подробное описание	51
6.11.2	Функции	51
6.11.2.1	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	51
6.11.2.2	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)	51
6.11.2.3	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	52
6.11.2.4	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	53
6.12	Конфигурация прерываний	54
6.12.1	Подробное описание	54
6.13	Цифровые компараторы	55
6.13.1	Подробное описание	55
6.13.2	Функции	55
6.13.2.1	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	55
6.13.2.2	ADC_DC_ITConfig(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef ADC_DC_Condition)	56
6.13.2.3	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	57
6.13.2.4	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	57
6.13.2.5	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	57
6.13.2.6	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	58
6.13.2.7	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_Module)	58
6.14	Секвенсоры	59
6.14.1	Подробное описание	59
6.14.2	Функции	59

6.14.2.1	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module)	59
6.14.2.2	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ Module, FunctionalState State)	59
6.14.2.3	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_↵ ITCountSEQRst)	60
6.14.2.4	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module)	60
6.14.2.5	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_S↵ EQ_Module)	60
6.14.2.6	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module)	61
6.14.2.7	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SE↵ Q_Module)	61
6.15	Константы	62
6.15.1	Подробное описание	62
6.16	Основная область флеш	63
6.16.1	Подробное описание	63
6.16.2	Макросы	63
6.16.2.1	BOOTFLASH_PAGE_SIZE_BYTES	63
6.16.2.2	BOOTFLASH_PAGE_TOTAL	63
6.16.2.3	BOOTFLASH_TOTAL_BYTES	63
6.17	Информационная область флеш	64
6.17.1	Подробное описание	64
6.17.2	Макросы	64
6.17.2.1	BOOTFLASH_INFO_PAGE_SIZE_BYTES	64
6.17.2.2	BOOTFLASH_INFO_PAGE_TOTAL	64
6.17.2.3	BOOTFLASH_INFO_TOTAL_BYTES	64
6.18	Типы	65
6.18.1	Подробное описание	65
6.18.2	Макросы	65
6.18.2.1	IS_BOOTFLASH_STATUS	65
6.18.3	Перечисления	65
6.18.3.1	BOOTFLASH_Status_TypeDef	65
6.19	Функции	66
6.19.1	Подробное описание	66
6.19.2	Функции	66
6.19.2.1	BOOTFLASH_Init(uint32_t SysClkFreq)	66
6.19.2.2	BOOTFLASH_ITCmd(FunctionalState State)	66
6.19.2.3	BOOTFLASH_OperationStatus()	66
6.19.2.4	BOOTFLASH_OperationStatusClear()	67

6.20	Основная область флеш	68
6.20.1	Подробное описание	68
6.20.2	Функции	68
6.20.2.1	BOOTFLASH_FullErase()	68
6.20.2.2	BOOTFLASH_PageErase(uint32_t PageNum)	68
6.20.2.3	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	68
6.21	Информационная область флеш	70
6.21.1	Подробное описание	70
6.21.2	Функции	70
6.21.2.1	BOOTFLASH_Info_PageErase(uint32_t PageNum)	70
6.21.2.2	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	70
6.22	Типы	71
6.22.1	Подробное описание	72
6.22.2	Макросы	72
6.22.2.1	IS_CAP_CAPTURE_MODE	72
6.22.2.2	IS_CAP_CAPTURE_POLARITY	72
6.22.2.3	IS_CAP_HALT	72
6.22.2.4	IS_CAP_MODE	72
6.22.2.5	IS_CAP_PWM_POLARITY	73
6.22.2.6	IS_CAP_SYNC_OUT	73
6.22.3	Перечисления	73
6.22.3.1	CAP_Capture_Mode_TypeDef	73
6.22.3.2	CAP_Capture_Polarity_TypeDef	73
6.22.3.3	CAP_Halt_TypeDef	73
6.22.3.4	CAP_Mode_TypeDef	74
6.22.3.5	CAP_PWM_Polarity_TypeDef	74
6.22.3.6	CAP_SyncOut_TypeDef	74
6.23	Константы	75
6.23.1	Подробное описание	75
6.24	Маски источников прерываний	76
6.24.1	Подробное описание	76
6.24.2	Макросы	76
6.24.2.1	CAP_ITSource_All	76
6.24.2.2	CAP_ITSource_CapEvent0	76
6.24.2.3	CAP_ITSource_CapEvent1	76
6.24.2.4	CAP_ITSource_CapEvent2	76
6.24.2.5	CAP_ITSource_CapEvent3	76
6.24.2.6	CAP_ITSource_GeneralInt	77

6.24.2.7	CAP_ITSource_TimerEqCompare	77
6.24.2.8	CAP_ITSource_TimerEqPeriod	77
6.24.2.9	CAP_ITSource_TimerOvf	77
6.24.2.10	IS_CAP_IT_SOURCE_SINGLE	77
6.25	Функции	78
6.25.1	Подробное описание	78
6.26	Конфигурация	79
6.26.1	Подробное описание	79
6.26.2	Функции	79
6.26.2.1	CAP_DeInit(NT_CAP_TypeDef *CAPx)	79
6.26.2.2	CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx)	79
6.26.2.3	CAP_GetTimer(NT_CAP_TypeDef *CAPx)	80
6.26.2.4	CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_↵ InitStruct)	80
6.26.2.5	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	80
6.26.2.6	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	81
6.26.2.7	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct)	82
6.26.2.8	CAP_SwSync(NT_CAP_TypeDef *CAPx)	82
6.26.2.9	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	82
6.26.2.10	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	83
6.27	Режим ШИМ	84
6.27.1	Подробное описание	84
6.27.2	Функции	84
6.27.2.1	CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx)	84
6.27.2.2	CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx)	84
6.27.2.3	CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx)	85
6.27.2.4	CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx)	85
6.27.2.5	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_↵ TypeDef *CAP_PWM_InitStruct)	85
6.27.2.6	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_↵ t CompareVal)	86
6.27.2.7	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)	87
6.27.2.8	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_↵ t CompareVal)	87
6.27.2.9	CAP_PWM_SetShadowPeriod(NT_CAP_TypeDef *CAPx, uint32_↵ t PeriodVal)	87
6.27.2.10	CAP_PWM_StructInit(CAP_PWM_Init_TypeDef *CAP_PWM_↵ InitStruct)	88
6.28	Режим захвата	89
6.28.1	Подробное описание	89
6.28.2	Функции	89
6.28.2.1	CAP_Capture_Cmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	89

6.28.2.2	CAP_Capture_GetCap0(NT_CAP_TypeDef *CAPx)	89
6.28.2.3	CAP_Capture_GetCap1(NT_CAP_TypeDef *CAPx)	90
6.28.2.4	CAP_Capture_GetCap2(NT_CAP_TypeDef *CAPx)	90
6.28.2.5	CAP_Capture_GetCap3(NT_CAP_TypeDef *CAPx)	90
6.28.2.6	CAP_Capture_Init(NT_CAP_TypeDef *CAPx, CAP_Capture_Init↔ _TypeDef *CAP_Capture_InitStruct)	91
6.28.2.7	CAP_Capture_SetCap0(NT_CAP_TypeDef *CAPx, uint32_t Value) .	92
6.28.2.8	CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value) .	92
6.28.2.9	CAP_Capture_SetCap2(NT_CAP_TypeDef *CAPx, uint32_t Value) .	92
6.28.2.10	CAP_Capture_SetCap3(NT_CAP_TypeDef *CAPx, uint32_t Value) .	93
6.28.2.11	CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_↔ Capture_InitStruct)	93
6.29	Прерывания	94
6.29.1	Подробное описание	94
6.29.2	Функции	94
6.29.2.1	CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)	94
6.29.2.2	CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_IT↔ Source)	94
6.29.2.3	CAP_ITPendClear(NT_CAP_TypeDef *CAPx)	95
6.29.2.4	CAP_ITPendStatus(NT_CAP_TypeDef *CAPx)	95
6.29.2.5	CAP_ITStatus(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	95
6.29.2.6	CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_IT↔ Source)	95
6.30	Константы	97
6.30.1	Подробное описание	97
6.31	Маски для CHANNEL_CFG	98
6.31.1	Подробное описание	98
6.31.2	Макросы	98
6.31.2.1	CHANNEL_CFG_CYCLE_CTRL_Msk	98
6.31.2.2	CHANNEL_CFG_CYCLE_CTRL_Pos	98
6.31.2.3	CHANNEL_CFG_DST_INC_Msk	98
6.31.2.4	CHANNEL_CFG_DST_INC_Pos	98
6.31.2.5	CHANNEL_CFG_DST_PROT_CTRL_Msk	99
6.31.2.6	CHANNEL_CFG_DST_PROT_CTRL_Pos	99
6.31.2.7	CHANNEL_CFG_DST_SIZE_Msk	99
6.31.2.8	CHANNEL_CFG_DST_SIZE_Pos	99
6.31.2.9	CHANNEL_CFG_N_MINUS_1_Msk	99
6.31.2.10	CHANNEL_CFG_N_MINUS_1_Pos	99
6.31.2.11	CHANNEL_CFG_NEXT_USEBURST_Msk	99
6.31.2.12	CHANNEL_CFG_NEXT_USEBURST_Pos	99

6.31.2.13	CHANNEL_CFG_R_POWER_Msk	99
6.31.2.14	CHANNEL_CFG_R_POWER_Pos	100
6.31.2.15	CHANNEL_CFG_SRC_INC_Msk	100
6.31.2.16	CHANNEL_CFG_SRC_INC_Pos	100
6.31.2.17	CHANNEL_CFG_SRC_PROT_CTRL_Msk	100
6.31.2.18	CHANNEL_CFG_SRC_PROT_CTRL_Pos	100
6.31.2.19	CHANNEL_CFG_SRC_SIZE_Msk	100
6.31.2.20	CHANNEL_CFG_SRC_SIZE_Pos	100
6.32	Маски каналов DMA	101
6.32.1	Подробное описание	101
6.32.2	Макросы	101
6.32.2.1	DMA_Channel_All	101
6.32.2.2	IS_GET_DMA_CHANNEL	101
6.33	Маски каналов по имени	103
6.33.1	Подробное описание	103
6.33.2	Макросы	103
6.33.2.1	DMA_Channel_ADCSEQ0	103
6.33.2.2	DMA_Channel_ADCSEQ1	103
6.33.2.3	DMA_Channel_ADCSEQ2	103
6.33.2.4	DMA_Channel_ADCSEQ3	103
6.33.2.5	DMA_Channel_ADCSEQ4	104
6.33.2.6	DMA_Channel_ADCSEQ5	104
6.33.2.7	DMA_Channel_ADCSEQ6	104
6.33.2.8	DMA_Channel_ADCSEQ7	104
6.33.2.9	DMA_Channel_SPI0_RX	104
6.33.2.10	DMA_Channel_SPI0_TX	104
6.33.2.11	DMA_Channel_SPI1_RX	104
6.33.2.12	DMA_Channel_SPI1_TX	104
6.33.2.13	DMA_Channel_SPI2_RX	104
6.33.2.14	DMA_Channel_SPI2_TX	104
6.33.2.15	DMA_Channel_SPI3_RX	105
6.33.2.16	DMA_Channel_SPI3_TX	105
6.33.2.17	DMA_Channel_UART0_RX	105
6.33.2.18	DMA_Channel_UART0_TX	105
6.33.2.19	DMA_Channel_UART1_RX	105
6.33.2.20	DMA_Channel_UART1_TX	105
6.33.2.21	DMA_Channel_UART2_RX	105
6.33.2.22	DMA_Channel_UART2_TX	105
6.33.2.23	DMA_Channel_UART3_RX	105
6.33.2.24	DMA_Channel_UART3_TX	106

6.34 Маски каналов по номеру	107
6.34.1 Подробное описание	107
6.34.2 Макросы	107
6.34.2.1 DMA_Channel_0	107
6.34.2.2 DMA_Channel_1	107
6.34.2.3 DMA_Channel_10	107
6.34.2.4 DMA_Channel_11	107
6.34.2.5 DMA_Channel_12	108
6.34.2.6 DMA_Channel_13	108
6.34.2.7 DMA_Channel_14	108
6.34.2.8 DMA_Channel_15	108
6.34.2.9 DMA_Channel_16	108
6.34.2.10 DMA_Channel_17	108
6.34.2.11 DMA_Channel_18	108
6.34.2.12 DMA_Channel_19	108
6.34.2.13 DMA_Channel_2	108
6.34.2.14 DMA_Channel_20	108
6.34.2.15 DMA_Channel_21	109
6.34.2.16 DMA_Channel_22	109
6.34.2.17 DMA_Channel_23	109
6.34.2.18 DMA_Channel_3	109
6.34.2.19 DMA_Channel_4	109
6.34.2.20 DMA_Channel_5	109
6.34.2.21 DMA_Channel_6	109
6.34.2.22 DMA_Channel_7	109
6.34.2.23 DMA_Channel_8	109
6.34.2.24 DMA_Channel_9	110
6.35 Типы	111
6.35.1 Подробное описание	112
6.35.2 Макросы	112
6.35.2.1 IS_DMA_ARBITRATION_RATE	112
6.35.2.2 IS_DMA_DATA_INC	112
6.35.2.3 IS_DMA_DATA_SIZE	113
6.35.2.4 IS_DMA_MODE	113
6.35.2.5 IS_DMA_STATE	113
6.35.3 Перечисления	113
6.35.3.1 DMA_ArbitrationRate_TypeDef	113
6.35.3.2 DMA_DataInc_TypeDef	114
6.35.3.3 DMA_DataSize_TypeDef	114
6.35.3.4 DMA_Mode_TypeDef	114

6.35.3.5	DMA_State_TypeDef	115
6.36	Функции	116
6.36.1	Подробное описание	116
6.37	Инициализация каналов DMA	117
6.37.1	Подробное описание	117
6.37.2	Функции	117
6.37.2.1	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	117
6.37.2.2	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	117
6.37.2.3	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	118
6.38	Инициализация контроллера DMA	119
6.38.1	Подробное описание	119
6.38.2	Функции	119
6.38.2.1	DMA_DeInit()	119
6.38.2.2	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	119
6.38.2.3	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	119
6.39	Конфигурация контроллера DMA	121
6.39.1	Подробное описание	121
6.39.2	Функции	121
6.39.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	121
6.39.2.2	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	121
6.39.2.3	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	122
6.39.2.4	DMA_MasterEnableCmd(FunctionalState State)	122
6.39.2.5	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	122
6.39.2.6	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	123
6.39.2.7	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	123
6.39.2.8	DMA_SWRequestCmd(uint32_t DMA_Channel)	123
6.39.2.9	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	124
6.40	Статусная информация	125
6.40.1	Подробное описание	125
6.40.2	Функции	125
6.40.2.1	DMA_ClearErrorStatus()	125
6.40.2.2	DMA_ErrorStatus()	125
6.40.2.3	DMA_MasterEnableStatus()	125
6.40.2.4	DMA_StateStatus()	125
6.40.2.5	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	126
6.41	Константы	127
6.41.1	Подробное описание	127
6.42	Маски адреса	128

6.42.1	Подробное описание	128
6.42.2	Макросы	128
6.42.2.1	EXTMEM_CEMask_Addr_11	128
6.42.2.2	EXTMEM_CEMask_Addr_11_19	128
6.42.2.3	EXTMEM_CEMask_Addr_12	128
6.42.2.4	EXTMEM_CEMask_Addr_13	128
6.42.2.5	EXTMEM_CEMask_Addr_14	128
6.42.2.6	EXTMEM_CEMask_Addr_15	128
6.42.2.7	EXTMEM_CEMask_Addr_16	129
6.42.2.8	EXTMEM_CEMask_Addr_17	129
6.42.2.9	EXTMEM_CEMask_Addr_18	129
6.42.2.10	EXTMEM_CEMask_Addr_19	129
6.43	Типы	130
6.43.1	Подробное описание	130
6.43.2	Макросы	130
6.43.2.1	IS_EXTMEM_READ_WAITSTATE	130
6.43.2.2	IS_EXTMEM_RW_WAITSTATE	131
6.43.2.3	IS_EXTMEM_WIDTH	131
6.43.2.4	IS_EXTMEM_WRITE_WAITSTATE	131
6.43.3	Перечисления	132
6.43.3.1	EXTMEM_ReadWaitState_TypeDef	132
6.43.3.2	EXTMEM_RWWaitState_TypeDef	132
6.43.3.3	EXTMEM_Width_TypeDef	132
6.43.3.4	EXTMEM_WriteWaitState_TypeDef	133
6.44	Функции	134
6.44.1	Подробное описание	134
6.44.2	Функции	134
6.44.2.1	EXTMEM_DeInit()	134
6.44.2.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	134
6.44.2.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	134
6.45	Типы	136
6.45.1	Подробное описание	137
6.45.2	Макросы	137
6.45.2.1	IS_GPIO_ALT_FUNC	137
6.45.2.2	IS_GPIO_DIR	137
6.45.2.3	IS_GPIO_INT_POL	137
6.45.2.4	IS_GPIO_INT_TYPE	138
6.45.2.5	IS_GPIO_LOAD	138
6.45.2.6	IS_GPIO_MODE	138
6.45.2.7	IS_GPIO_OUT	138

6.45.2.8	IS_GPIO_OUT_MODE	138
6.45.2.9	IS_GPIO_PULLUP	139
6.45.2.10	IS_GPIO_QUAL	139
6.45.2.11	IS_GPIO_QUAL_MODE	139
6.45.2.12	IS_GPIO_SYNC	139
6.45.3	Перечисления	139
6.45.3.1	BitAction	139
6.45.3.2	GPIO_AltFunc_TypeDef	140
6.45.3.3	GPIO_Dir_TypeDef	140
6.45.3.4	GPIO_IntPol_TypeDef	140
6.45.3.5	GPIO_IntType_TypeDef	140
6.45.3.6	GPIO_Load_TypeDef	140
6.45.3.7	GPIO_Mode_TypeDef	141
6.45.3.8	GPIO_Out_TypeDef	141
6.45.3.9	GPIO_OutMode_TypeDef	141
6.45.3.10	GPIO_PullUp_TypeDef	141
6.45.3.11	GPIO_Qual_TypeDef	141
6.45.3.12	GPIO_QualMode_TypeDef	142
6.45.3.13	GPIO_Sync_TypeDef	142
6.46	Константы	143
6.46.1	Подробное описание	143
6.47	Маски пинов	144
6.47.1	Подробное описание	144
6.47.2	Макросы	144
6.47.2.1	GPIO_Pin_0	144
6.47.2.2	GPIO_Pin_0_3	144
6.47.2.3	GPIO_Pin_0_7	144
6.47.2.4	GPIO_Pin_1	145
6.47.2.5	GPIO_Pin_10	145
6.47.2.6	GPIO_Pin_11	145
6.47.2.7	GPIO_Pin_12	145
6.47.2.8	GPIO_Pin_12_15	145
6.47.2.9	GPIO_Pin_13	145
6.47.2.10	GPIO_Pin_14	145
6.47.2.11	GPIO_Pin_15	145
6.47.2.12	GPIO_Pin_2	145
6.47.2.13	GPIO_Pin_3	145
6.47.2.14	GPIO_Pin_4	146
6.47.2.15	GPIO_Pin_4_7	146
6.47.2.16	GPIO_Pin_5	146

6.47.2.17	GPIO_Pin_6	146
6.47.2.18	GPIO_Pin_7	146
6.47.2.19	GPIO_Pin_8	146
6.47.2.20	GPIO_Pin_8_11	146
6.47.2.21	GPIO_Pin_8_15	146
6.47.2.22	GPIO_Pin_9	146
6.47.2.23	GPIO_Pin_All	147
6.47.2.24	IS_GET_GPIO_PIN	147
6.48	Функции	148
6.48.1	Подробное описание	148
6.49	Инициализация и деинициализация	149
6.49.1	Подробное описание	149
6.49.2	Функции	149
6.49.2.1	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	149
6.49.2.2	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	149
6.49.2.3	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	150
6.50	Чтение и запись	151
6.50.1	Подробное описание	151
6.51	Чтение	152
6.51.1	Подробное описание	152
6.51.2	Функции	152
6.51.2.1	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	152
6.51.2.2	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	152
6.51.2.3	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	152
6.52	Запись	154
6.52.1	Подробное описание	154
6.52.2	Функции	154
6.52.2.1	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	154
6.52.2.2	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	154
6.52.2.3	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	154
6.53	Битовые операции	156
6.53.1	Подробное описание	156
6.53.2	Функции	156
6.53.2.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	156
6.53.2.2	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	156
6.53.2.3	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	156
6.54	Фильтрация	158
6.54.1	Подробное описание	158

6.54.2	Функции	158
6.54.2.1	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	158
6.54.2.2	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	158
6.54.2.3	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	159
6.55	Прерывания	160
6.55.1	Подробное описание	160
6.55.2	Функции	160
6.55.2.1	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	160
6.55.2.2	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	160
6.55.2.3	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	161
6.56	Константы	162
6.56.1	Подробное описание	162
6.56.2	Макросы	162
6.56.2.1	RCC_CLK_CHANGE_TIMEOUT	162
6.56.2.2	RCC_CLK_PLL_STABLE_TIMEOUT	162
6.57	Типы	163
6.57.1	Подробное описание	164
6.57.2	Макросы	165
6.57.2.1	IS_RCC_ADC_CLK	165
6.57.2.2	IS_RCC_PERIPH_CLK	165
6.57.2.3	IS_RCC_PLL_NO	165
6.57.2.4	IS_RCC_PLL_REF	165
6.57.2.5	IS_RCC_SPI_CLK	166
6.57.2.6	IS_RCC_SYS_CLK	166
6.57.2.7	IS_RCC_UART_CLK	166
6.57.2.8	IS_RCC_USB_CLK	166
6.57.2.9	IS_RCC_USB_FREQ	167
6.57.3	Перечисления	167
6.57.3.1	RCC_ADCClk_TypeDef	167
6.57.3.2	RCC_PeriphClk_TypeDef	167
6.57.3.3	RCC_PeriphRst_TypeDef	168
6.57.3.4	RCC_PLLNO_TypeDef	169
6.57.3.5	RCC_PLLRef_TypeDef	169
6.57.3.6	RCC_SPIClk_TypeDef	169
6.57.3.7	RCC_SysClk_TypeDef	170
6.57.3.8	RCC_UARTClk_TypeDef	170

6.57.3.9	RCC_USBClk_TypeDef	170
6.57.3.10	RCC_USBFreq_TypeDef	170
6.58	Функции	171
6.58.1	Подробное описание	171
6.58.2	Функции	171
6.58.2.1	RCC_SysClkDiv2Out(FunctionalState State)	171
6.59	Конфигурация PLL	172
6.59.1	Подробное описание	172
6.59.2	Функции	172
6.59.2.1	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	172
6.59.2.2	RCC_PLLDeInit()	172
6.59.2.3	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	173
6.59.2.4	RCC_PLLPowerDownCmd(FunctionalState State)	174
6.59.2.5	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	174
6.60	Управление тактированием	175
6.60.1	Подробное описание	175
6.60.2	Функции	175
6.60.2.1	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	175
6.60.2.2	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	175
6.60.2.3	RCC_SysClkStatus()	176
6.61	Тактирование USB	177
6.61.1	Подробное описание	177
6.61.2	Функции	177
6.61.2.1	RCC_USBClkCmd(FunctionalState State)	177
6.61.2.2	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	177
6.62	Тактирование UART	178
6.62.1	Подробное описание	178
6.62.2	Функции	178
6.62.2.1	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	178
6.62.2.2	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	178
6.62.2.3	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	179
6.63	Тактирование SPI	181
6.63.1	Подробное описание	181
6.63.2	Функции	181
6.63.2.1	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	181

6.63.2.2	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	181
6.63.2.3	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	181
6.64	Тактирование ADC	183
6.64.1	Подробное описание	183
6.64.2	Функции	183
6.64.2.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	183
6.64.2.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	183
6.65	Управление сбросом	184
6.65.1	Подробное описание	184
6.65.2	Функции	184
6.65.2.1	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	184
6.66	Типы	185
6.66.1	Подробное описание	186
6.66.2	Макросы	186
6.66.2.1	IS_RTC_FORMAT	186
6.66.2.2	IS_RTC_MONTH	186
6.66.2.3	IS_RTC_WEEKDAY	186
6.66.3	Перечисления	186
6.66.3.1	RTC_Format_TypeDef	186
6.66.3.2	RTC_Month_TypeDef	187
6.66.3.3	RTC_Weekday_TypeDef	187
6.67	Функции	188
6.67.1	Подробное описание	188
6.68	Типы	189
6.68.1	Подробное описание	189
6.68.2	Макросы	189
6.68.2.1	IS_TIMER_EXT_INPUT	189
6.68.3	Перечисления	189
6.68.3.1	TIMER_ExtInput_TypeDef	189
6.69	Константы	190
6.70	Функции	191
6.70.1	Подробное описание	191
6.71	Конфигурация	192
6.71.1	Подробное описание	192
6.71.2	Функции	192
6.71.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	192

6.71.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	192
6.71.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	193
6.71.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	193
6.71.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	193
6.71.2.6	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	194
6.71.2.7	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	194
6.71.2.8	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	194
6.72	Прерывания	196
6.72.1	Подробное описание	196
6.72.2	Функции	196
6.72.2.1	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	196
6.72.2.2	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	196
6.72.2.3	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	196
6.73	Типы	198
6.73.1	Подробное описание	199
6.73.2	Макросы	199
6.73.2.1	IS_UART_DATA_WIDTH	199
6.73.2.2	IS_UART_DIR	199
6.73.2.3	IS_UART_ERROR	200
6.73.2.4	IS_UART_FIFO_LEVEL	200
6.73.2.5	IS_UART_FLAG	200
6.73.2.6	IS_UART_GET_IT_SOURCE	200
6.73.2.7	IS_UART_PARITY_BIT	201
6.73.2.8	IS_UART_STOP_BIT	201
6.73.3	Перечисления	201
6.73.3.1	UART_DataWidth_TypeDef	201
6.73.3.2	UART_Dir_Typedef	202
6.73.3.3	UART_Error_Typedef	202
6.73.3.4	UART_FIFOLevel_TypeDef	202
6.73.3.5	UART_Flag_Typedef	202
6.73.3.6	UART_ITSource_Typedef	203
6.73.3.7	UART_ParityBit_TypeDef	203
6.73.3.8	UART_StopBit_TypeDef	203
6.74	Константы	204
6.75	Функции	205
6.75.1	Подробное описание	205

6.75.2	Функции	205
6.75.2.1	UART_BaudRateDivConfig(<code>NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv</code>)	205
6.75.2.2	UART_Break(<code>NT_UART_TypeDef *UARTx, FunctionalState State</code>)	205
6.75.2.3	UART_Cmd(<code>NT_UART_TypeDef *UARTx, FunctionalState State</code>)	206
6.76	Инициализация и деинициализация	207
6.76.1	Подробное описание	207
6.76.2	Функции	207
6.76.2.1	UART_DeInit(<code>NT_UART_TypeDef *UARTx</code>)	207
6.76.2.2	UART_Init(<code>NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct</code>)	207
6.76.2.3	UART_StructInit(<code>UART_Init_TypeDef *UART_InitStruct</code>)	208
6.77	Прием и передача	210
6.77.1	Подробное описание	210
6.77.2	Функции	210
6.77.2.1	UART_ErrorStatus(<code>NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error</code>)	210
6.77.2.2	UART_ErrorStatusClear(<code>NT_UART_TypeDef *UARTx</code>)	210
6.77.2.3	UART_FlagStatus(<code>NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag</code>)	211
6.77.2.4	UART_RecieveData(<code>NT_UART_TypeDef *UARTx</code>)	212
6.77.2.5	UART_SendData(<code>NT_UART_TypeDef *UARTx, uint32_t Data</code>)	212
6.78	Режим модема	213
6.78.1	Подробное описание	213
6.78.2	Функции	213
6.78.2.1	UART_ModemConfig(<code>NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct</code>)	213
6.78.2.2	UART_ModemStructInit(<code>UART_ModemInit_TypeDef *UART_ModemInitStruct</code>)	213
6.79	Прерывания	214
6.79.1	Подробное описание	214
6.79.2	Функции	214
6.79.2.1	UART_ITCmd(<code>NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State</code>)	214
6.79.2.2	UART_ITFIFOLevelConfig(<code>NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel</code>)	214
6.79.2.3	UART_ITMaskedStatus(<code>NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource</code>)	215
6.79.2.4	UART_ITRawStatus(<code>NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource</code>)	215
6.79.2.5	UART_ITStatusClear(<code>NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource</code>)	215
6.80	Настройка DMA	217

6.80.1	Подробное описание	217
6.80.2	Функции	217
6.80.2.1	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	217
6.80.2.2	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	217
6.81	Константы	218
6.81.1	Подробное описание	218
6.82	Основная область флеш	219
6.82.1	Подробное описание	219
6.82.2	Макросы	219
6.82.2.1	USERFLASH_PAGE_SIZE_BYTES	219
6.82.2.2	USERFLASH_PAGE_TOTAL	219
6.82.2.3	USERFLASH_TOTAL_BYTES	219
6.83	Информационная область флеш	220
6.83.1	Подробное описание	220
6.83.2	Макросы	220
6.83.2.1	USERFLASH_INFO_PAGE_SIZE_BYTES	220
6.83.2.2	USERFLASH_INFO_PAGE_TOTAL	220
6.83.2.3	USERFLASH_INFO_TOTAL_BYTES	220
6.84	Типы	221
6.84.1	Подробное описание	221
6.84.2	Макросы	221
6.84.2.1	IS_USERFLASH_STATUS	221
6.84.3	Перечисления	221
6.84.3.1	USERFLASH_Status_TypeDef	221
6.85	Функции	222
6.85.1	Подробное описание	222
6.85.2	Функции	222
6.85.2.1	USERFLASH_Init(uint32_t SysClkFreq)	222
6.85.2.2	USERFLASH_ITCmd(FunctionalState State)	222
6.85.2.3	USERFLASH_OperationStatus()	222
6.85.2.4	USERFLASH_OperationStatusClear()	223
6.86	Основная область флеш	224
6.86.1	Подробное описание	224
6.86.2	Функции	224
6.86.2.1	USERFLASH_FullErase()	224
6.86.2.2	USERFLASH_PageErase(uint32_t PageNum)	224
6.86.2.3	USERFLASH_Read(uint32_t Address)	224
6.86.2.4	USERFLASH_Write(uint32_t Address, uint32_t Data)	225

6.87 Информационная область флеш	226
6.87.1 Подробное описание	226
6.87.2 Функции	226
6.87.2.1 USERFLASH_Info_PageErase(uint32_t PageNum)	226
6.87.2.2 USERFLASH_Info_Read(uint32_t Address)	226
6.87.2.3 USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	226
6.88 Типы	228
6.88.1 Подробное описание	228
6.89 Константы	229
6.90 Функции	230
6.90.1 Подробное описание	230
6.91 Конфигурация	231
6.91.1 Подробное описание	231
6.91.2 Функции	231
6.91.2.1 WATCHDOG_Cmd(FunctionalState State)	231
6.91.2.2 WATCHDOG_GetCounter()	231
6.91.2.3 WATCHDOG_GetReload()	231
6.91.2.4 WATCHDOG_LockCmd(FunctionalState State)	232
6.91.2.5 WATCHDOG_RstCmd(FunctionalState State)	232
6.91.2.6 WATCHDOG_SetReload(uint32_t ReloadVal)	232
6.92 Прерывания	233
6.92.1 Подробное описание	233
6.92.2 Функции	233
6.92.2.1 WATCHDOG_ITMaskedStatus()	233
6.92.2.2 WATCHDOG_ITRawStatus()	233
6.92.2.3 WATCHDOG_ITStatusClear()	233
6.93 ADC	234
6.93.1 Подробное описание	234
6.94 Приватные данные	235
6.94.1 Подробное описание	235
6.95 Приватные константы	236
6.95.1 Подробное описание	236
6.96 Начальные значения регистров	237
6.97 Приватные функции	238
6.97.1 Подробное описание	240
6.97.2 Функции	240
6.97.2.1 ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	240
6.97.2.2 ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	240

6.97.2.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module) . .	240
6.97.2.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	240
6.97.2.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↔ DC_DC_Init_TypeDef *ADC_DC_InitStruct)	241
6.97.2.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	241
6.97.2.7	ADC_DC_ITConfig(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition↔ _TypeDef ADC_DC_Condition)	241
6.97.2.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	242
6.97.2.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	242
6.97.2.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	242
6.97.2.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	243
6.97.2.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	243
6.97.2.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	243
6.97.2.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	244
6.97.2.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	244
6.97.2.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	244
6.97.2.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↔ Def *ADC_InitStruct)	245
6.97.2.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	246
6.97.2.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	246
6.97.2.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State)	246
6.97.2.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ↔ _Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel) .	247
6.97.2.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↔ SEQ_Module)	247
6.97.2.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef A↔ DC_SEQ_Module)	247
6.97.2.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC↔ _SEQ_Module)	248
6.97.2.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	248
6.97.2.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_S↔ EQ_Module)	248
6.97.2.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef AD↔ C_SEQ_Module)	249

6.97.2.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	249
6.97.2.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	249
6.97.2.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	250
6.97.2.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	251
6.97.2.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	251
6.97.2.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	251
6.97.2.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	252
6.97.2.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	252
6.97.2.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	252
6.97.2.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	253
6.97.2.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	253
6.97.2.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	253
6.97.2.40	ADC_SEQ_SWReq()	254
6.97.2.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	254
6.98	BOOTFLASH	255
6.98.1	Подробное описание	255
6.99	Приватные данные	256
6.99.1	Подробное описание	256
6.100	Приватные константы	257
6.101	Приватные функции	258
6.101.1	Подробное описание	258
6.101.2	Функции	258
6.101.2.1	BOOTFLASH_FullErase()	258
6.101.2.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	258
6.101.2.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	259
6.101.2.4	BOOTFLASH_Init(uint32_t SysClkFreq)	259
6.101.2.5	BOOTFLASH_ITCmd(FunctionalState State)	259
6.101.2.6	BOOTFLASH_OperationStatus()	259
6.101.2.7	BOOTFLASH_OperationStatusClear()	260
6.101.2.8	BOOTFLASH_PageErase(uint32_t PageNum)	260

6.101.2.9 BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	260
6.102CAP	261
6.102.1 Подробное описание	261
6.103Приватные данные	262
6.103.1 Подробное описание	262
6.104Приватные константы	263
6.105Приватные функции	264
6.105.1 Подробное описание	265
6.105.2 Функции	265
6.105.2.1 CAP_Capture_Cmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	265
6.105.2.2 CAP_Capture_GetCap0(NT_CAP_TypeDef *CAPx)	265
6.105.2.3 CAP_Capture_GetCap1(NT_CAP_TypeDef *CAPx)	266
6.105.2.4 CAP_Capture_GetCap2(NT_CAP_TypeDef *CAPx)	266
6.105.2.5 CAP_Capture_GetCap3(NT_CAP_TypeDef *CAPx)	266
6.105.2.6 CAP_Capture_Init(NT_CAP_TypeDef *CAPx, CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	267
6.105.2.7 CAP_Capture_SetCap0(NT_CAP_TypeDef *CAPx, uint32_t Value)	268
6.105.2.8 CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value)	268
6.105.2.9 CAP_Capture_SetCap2(NT_CAP_TypeDef *CAPx, uint32_t Value)	268
6.105.2.10CAP_Capture_SetCap3(NT_CAP_TypeDef *CAPx, uint32_t Value)	269
6.105.2.11CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	269
6.105.2.12CAP_DeInit(NT_CAP_TypeDef *CAPx)	269
6.105.2.13CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx)	270
6.105.2.14CAP_GetTimer(NT_CAP_TypeDef *CAPx)	270
6.105.2.15CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_InitStruct)	270
6.105.2.16CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)	271
6.105.2.17CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	271
6.105.2.18CAP_ITPendClear(NT_CAP_TypeDef *CAPx)	271
6.105.2.19CAP_ITPendStatus(NT_CAP_TypeDef *CAPx)	271
6.105.2.20CAP_ITStatus(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	272
6.105.2.21CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	272
6.105.2.22CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx)	272
6.105.2.23CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx)	273
6.105.2.24CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx)	274
6.105.2.25CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx)	274

6.105.2.26	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_↵ TypeDef *CAP_PWM_InitStruct)	274
6.105.2.27	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_↵ t CompareVal)	275
6.105.2.28	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)	275
6.105.2.29	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_↵ t CompareVal)	275
6.105.2.30	CAP_PWM_SetShadowPeriod(NT_CAP_TypeDef *CAPx, uint32_↵ t PeriodVal)	275
6.105.2.31	CAP_PWM_StructInit(CAP_PWM_Init_TypeDef *CAP_PWM_↵ InitStruct)	276
6.105.2.32	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	276
6.105.2.33	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	276
6.105.2.34	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct)	277
6.105.2.35	CAP_SwSync(NT_CAP_TypeDef *CAPx)	277
6.105.2.36	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State) . .	277
6.105.2.37	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State) .	277
6.106	DMA	279
6.106.1	Подробное описание	279
6.107	Приватные данные	280
6.107.1	Подробное описание	280
6.108	Приватные константы	281
6.108.1	Подробное описание	281
6.109	Начальные значения регистров	282
6.110	Приватные функции	283
6.110.1	Подробное описание	283
6.110.2	Функции	283
6.110.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	283
6.110.2.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel) . . .	284
6.110.2.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	284
6.110.2.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_↵ ChannelInit_TypeDef *DMA_ChannelInitStruct)	284
6.110.2.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_↵ ChannelInitStruct)	285
6.110.2.6	DMA_ClearErrorStatus()	285
6.110.2.7	DMA_DeInit()	286
6.110.2.8	DMA_ErrorStatus()	287
6.110.2.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	287
6.110.2.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	287
6.110.2.11	DMA_MasterEnableCmd(FunctionalState State)	288
6.110.2.12	DMA_MasterEnableStatus()	288

6.110.2.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State) . .	288
6.110.2.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection) .	288
6.110.2.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State) .	289
6.110.2.16	DMA_StateStatus()	289
6.110.2.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	289
6.110.2.18	DMA_SWRequestCmd(uint32_t DMA_Channel)	290
6.110.2.19	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State) .	290
6.110.2.20	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	290
6.111	EXTMEM	291
6.111.1	Подробное описание	291
6.112	Приватные данные	292
6.112.1	Подробное описание	292
6.113	Приватные константы	293
6.113.1	Подробное описание	293
6.114	Начальные значения регистров	294
6.114.1	Подробное описание	294
6.114.2	Макросы	294
6.114.2.1	EXT_MEM_CFG_Reset_Value	294
6.115	Приватные функции	295
6.115.1	Подробное описание	295
6.115.2	Функции	295
6.115.2.1	EXTMEM_DeInit()	295
6.115.2.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct) . . .	295
6.115.2.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	295
6.116	GPIO	297
6.116.1	Подробное описание	297
6.117	Приватные данные	298
6.117.1	Подробное описание	298
6.118	Приватные константы	299
6.118.1	Подробное описание	299
6.119	Начальные значения регистров	300
6.119.1	Подробное описание	300
6.119.2	Макросы	300
6.119.2.1	GPIO_DATAOUT_Reset_Value	300
6.119.2.2	GPIO_GPIODEN0_Reset_Value	300
6.119.2.3	GPIO_GPIODEN1_Reset_Value	300
6.119.2.4	GPIO_GPIODEN2_Reset_Value	300
6.119.2.5	GPIO_GPIODEN3_Reset_Value	301
6.119.2.6	GPIO_GPIOODCTLx_Reset_Value	301
6.119.2.7	GPIO_GPIOODSCTLx_Reset_Value	301

6.119.2.8	GPIO_GPIOCTLx_Reset_Value	301
6.119.2.9	GPIO_GPIOPUCTLx_Reset_Value	301
6.119.2.10	GPIO_GPIOQEx_Reset_Value	301
6.119.2.11	GPIO_GPIOQMx_Reset_Value	301
6.119.2.12	GPIO_GPIOQPx_Reset_Value	301
6.119.2.13	GPIO_GPIOSEx_Reset_Value	302
6.120	Маски портов	303
6.120.1	Подробное описание	303
6.120.2	Макросы	303
6.120.2.1	GPIO_Regs_A_C_E_G_Mask	303
6.120.2.2	GPIO_Regs_B_D_F_H_Mask	303
6.120.2.3	GPIO_Regs_GPIOA_Mask	303
6.120.2.4	GPIO_Regs_GPIOB_Mask	303
6.120.2.5	GPIO_Regs_GPIOC_Mask	303
6.120.2.6	GPIO_Regs_GPIOD_Mask	303
6.120.2.7	GPIO_Regs_GPIOE_Mask	304
6.120.2.8	GPIO_Regs_GPIOF_Mask	304
6.120.2.9	GPIO_Regs_GPIOG_Mask	304
6.120.2.10	GPIO_Regs_GPIOH_Mask	304
6.121	Приватные функции	305
6.121.1	Подробное описание	305
6.121.2	Функции	305
6.121.2.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	305
6.121.2.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	306
6.121.2.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	306
6.121.2.4	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	307
6.121.2.5	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	308
6.121.2.6	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	308
6.121.2.7	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	308
6.121.2.8	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePerod)	309
6.121.2.9	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	309
6.121.2.10	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	309
6.121.2.11	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	310
6.121.2.12	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	310
6.121.2.13	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	310

6.121.2.14	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	311
6.121.2.15	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	311
6.121.2.16	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	311
6.121.2.17	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	312
6.121.2.18	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	312
6.122	RCC	313
6.122.1	Подробное описание	313
6.123	Приватные данные	314
6.123.1	Подробное описание	314
6.124	Приватные константы	315
6.124.1	Подробное описание	315
6.125	Начальные значения регистров	316
6.125.1	Подробное описание	316
6.125.2	Макросы	316
6.125.2.1	RCC_PLL_CTRL_Reset_Value	316
6.125.2.2	RCC_PLL_NF_Reset_Value	316
6.125.2.3	RCC_PLL_NR_Reset_Value	316
6.125.2.4	RCC_PLL_OD_Reset_Value	316
6.126	Приватные функции	317
6.126.1	Подробное описание	318
6.126.2	Функции	318
6.126.2.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	318
6.126.2.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	318
6.126.2.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	318
6.126.2.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	319
6.126.2.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	319
6.126.2.6	RCC_PLLDeInit()	320
6.126.2.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	320
6.126.2.8	RCC_PLLPowerDownCmd(FunctionalState State)	321
6.126.2.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	321
6.126.2.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	321
6.126.2.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	322
6.126.2.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	322

6.126.2.1	<code>RCC_SysClkDiv2Out(FunctionalState State)</code>	322
6.126.2.1	<code>RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)</code>	323
6.126.2.1	<code>RCC_SysClkStatus()</code>	323
6.126.2.1	<code>RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)</code>	323
6.126.2.1	<code>RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)</code>	324
6.126.2.1	<code>RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)</code>	324
6.126.2.1	<code>RCC_USBClkCmd(FunctionalState State)</code>	324
6.126.2.2	<code>RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)</code>	325
6.126.2.2	<code>RCC_WaitClkChange(RCC_SysClk_TypeDef RCC_SysClk)</code>	325
6.127	RTC	326
6.127.1	Подробное описание	326
6.128	Приватные данные	327
6.128.1	Подробное описание	327
6.129	Приватные функции	328
6.129.1	Подробное описание	328
6.130	TIMER	329
6.130.1	Подробное описание	329
6.131	Приватные данные	330
6.131.1	Подробное описание	330
6.132	Приватные константы	331
6.133	Приватные функции	332
6.133.1	Подробное описание	332
6.133.2	Функции	332
6.133.2.1	<code>TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)</code>	332
6.133.2.2	<code>TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)</code>	332
6.133.2.3	<code>TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)</code>	333
6.133.2.4	<code>TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)</code>	333
6.133.2.5	<code>TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)</code>	333
6.133.2.6	<code>TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)</code>	334
6.133.2.7	<code>TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)</code>	334
6.133.2.8	<code>TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)</code>	334
6.133.2.9	<code>TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)</code>	335
6.133.2.10	<code>TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)</code>	335

6.133.2.1	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	335
6.134	UART	336
6.134.1	Подробное описание	336
6.135	Приватные данные	337
6.135.1	Подробное описание	337
6.136	Приватные константы	338
6.137	Приватные функции	339
6.137.1	Подробное описание	340
6.137.2	Функции	340
6.137.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	340
6.137.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	340
6.137.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	340
6.137.2.4	UART_DeInit(NT_UART_TypeDef *UARTx)	341
6.137.2.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	341
6.137.2.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	341
6.137.2.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error)	342
6.137.2.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	342
6.137.2.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag)	342
6.137.2.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	342
6.137.2.11	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)	343
6.137.2.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	343
6.137.2.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	344
6.137.2.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	345
6.137.2.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	345
6.137.2.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	345
6.137.2.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	346
6.137.2.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	346
6.137.2.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	346
6.137.2.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	347

6.138	USERFLASH	349
6.138.1	Подробное описание	349
6.139	Приватные данные	350
6.139.1	Подробное описание	350
6.140	Приватные константы	351
6.141	Приватные функции	352
6.141.1	Подробное описание	352
6.141.2	Функции	352
6.141.2.1	USERFLASH_FullErase()	352
6.141.2.2	USERFLASH_Info_PageErase(uint32_t PageNum)	352
6.141.2.3	USERFLASH_Info_Read(uint32_t Address)	353
6.141.2.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	353
6.141.2.5	USERFLASH_Init(uint32_t SysClkFreq)	353
6.141.2.6	USERFLASH_ITCmd(FunctionalState State)	353
6.141.2.7	USERFLASH_OperationStatus()	354
6.141.2.8	USERFLASH_OperationStatusClear()	354
6.141.2.9	USERFLASH_PageErase(uint32_t PageNum)	354
6.141.2.10	USERFLASH_Read(uint32_t Address)	354
6.141.2.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	355
6.142	WATCHDOG	356
6.142.1	Подробное описание	356
6.143	Приватные данные	357
6.143.1	Подробное описание	357
6.144	Приватные константы	358
6.144.1	Подробное описание	358
6.144.2	Макросы	358
6.144.2.1	WATCHDOG_Lock_Value	358
6.144.2.2	WATCHDOG_Unlock_Value	358
6.145	Приватные функции	359
6.145.1	Подробное описание	359
6.145.2	Функции	359
6.145.2.1	WATCHDOG_Cmd(FunctionalState State)	359
6.145.2.2	WATCHDOG_GetCounter()	359
6.145.2.3	WATCHDOG_GetReload()	359
6.145.2.4	WATCHDOG_ITMaskedStatus()	360
6.145.2.5	WATCHDOG_ITRawStatus()	360
6.145.2.6	WATCHDOG_ITStatusClear()	360
6.145.2.7	WATCHDOG_LockCmd(FunctionalState State)	360
6.145.2.8	WATCHDOG_RstCmd(FunctionalState State)	360
6.145.2.9	WATCHDOG_SetReload(uint32_t ReloadVal)	361

6.146	Драйвер периферии	362
6.146.1	Подробное описание	362
6.147	Настройка драйвера	363
6.147.1	Подробное описание	363
6.147.2	Макросы	363
6.147.2.1	EXT_OSC_VALUE	363
6.147.2.2	INT_OSC_VALUE	363
6.148	Макросы	364
6.148.1	Подробное описание	364
6.149	Типы	365
6.149.1	Подробное описание	365
6.149.2	Макросы	365
6.149.2.1	IS_CAP_ALL_PERIPH	365
6.149.2.2	IS_GPIO_ALL_PERIPH	366
6.149.2.3	IS_SPI_ALL_PERIPH	366
6.149.2.4	IS_TIMER_ALL_PERIPH	366
6.149.2.5	IS_UART_ALL_PERIPH	366
6.150	Периферия	368
6.150.1	Подробное описание	368
7	Структуры данных	369
7.1	Структура _CHANNEL_CFG_bits	369
7.1.1	Подробное описание	369
7.1.2	Поля	369
7.1.2.1	CYCLE_CTRL	369
7.1.2.2	DST_INC	370
7.1.2.3	DST_PROT_BUFFERABLE	370
7.1.2.4	DST_PROT_CACHEABLE	370
7.1.2.5	DST_PROT_PRIVILEGED	370
7.1.2.6	DST_SIZE	370
7.1.2.7	N_MINUS_1	370
7.1.2.8	NEXT_USEBURST	370
7.1.2.9	R_POWER	370
7.1.2.10	SRC_INC	371
7.1.2.11	SRC_PROT_BUFFERABLE	371
7.1.2.12	SRC_PROT_CACHEABLE	371
7.1.2.13	SRC_PROT_PRIVILEGED	371
7.1.2.14	SRC_SIZE	371
7.2	Структура ADC_DC_Init_TypeDef	371
7.2.1	Подробное описание	372

7.2.2	Поля	372
7.2.2.1	ADC_DC_Channel	372
7.2.2.2	ADC_DC_Condition	372
7.2.2.3	ADC_DC_Mode	372
7.2.2.4	ADC_DC_ThresholdHigh	372
7.2.2.5	ADC_DC_ThresholdLow	372
7.3	Структура ADC_Init_TypeDef	373
7.3.1	Подробное описание	373
7.3.2	Поля	373
7.3.2.1	ADC_Average	373
7.3.2.2	ADC_Measure_A	373
7.3.2.3	ADC_Measure_B	373
7.3.2.4	ADC_Mode	374
7.3.2.5	ADC_Phase	374
7.3.2.6	ADC_Resolution	374
7.4	Структура ADC_SEQ_Init_TypeDef	374
7.4.1	Подробное описание	374
7.4.2	Поля	374
7.4.2.1	ADC_Channels	374
7.4.2.2	ADC_SEQ_ConversionCount	375
7.4.2.3	ADC_SEQ_ConversionDelay	375
7.4.2.4	ADC_SEQ_DC	375
7.4.2.5	ADC_SEQ_StartEvent	375
7.4.2.6	ADC_SEQ_SWReqEn	375
7.5	Структура CAP_Capture_Init_TypeDef	375
7.5.1	Подробное описание	376
7.5.2	Поля	376
7.5.2.1	CAP_Capture_PolarityEvent0	376
7.5.2.2	CAP_Capture_PolarityEvent1	376
7.5.2.3	CAP_Capture_PolarityEvent2	376
7.5.2.4	CAP_Capture_PolarityEvent3	376
7.5.2.5	CAP_Capture_Prescale	377
7.5.2.6	CAP_Capture_RstEvent0	377
7.5.2.7	CAP_Capture_RstEvent1	377
7.5.2.8	CAP_Capture_RstEvent2	377
7.5.2.9	CAP_Capture_RstEvent3	377
7.5.2.10	CAP_Capture_StopVal	377
7.5.2.11	CAP_CaptureMode	377
7.6	Структура CAP_Init_TypeDef	378
7.6.1	Подробное описание	378

7.6.2	Поля	378
7.6.2.1	CAP_Halt	378
7.6.2.2	CAP_Mode	378
7.6.2.3	CAP_SyncCmd	378
7.6.2.4	CAP_SyncOut	379
7.7	Структура CAP_PWM_Init_TypeDef	379
7.7.1	Подробное описание	379
7.7.2	Поля	379
7.7.2.1	CAP_PWM_Compare	379
7.7.2.2	CAP_PWM_Period	379
7.7.2.3	CAP_PWM_Polarity	379
7.8	Структура DMA_Channel_TypeDef	380
7.8.1	Подробное описание	380
7.8.2	Поля	380
7.8.2.1	CHANNEL_CFG	380
7.8.2.2	CHANNEL_CFG_bit	380
7.8.2.3	DST_DATA_END	380
7.8.2.4	SRC_DATA_END	380
7.9	Структура DMA_ChannelInit_TypeDef	381
7.9.1	Подробное описание	381
7.9.2	Поля	381
7.9.2.1	DMA_ArbitrationRate	381
7.9.2.2	DMA_DstDataEndPtr	381
7.9.2.3	DMA_DstDataInc	382
7.9.2.4	DMA_DstDataSize	382
7.9.2.5	DMA_DstProtect	382
7.9.2.6	DMA_Mode	382
7.9.2.7	DMA_NextUseburst	382
7.9.2.8	DMA_SrcDataEndPtr	382
7.9.2.9	DMA_SrcDataInc	382
7.9.2.10	DMA_SrcDataSize	383
7.9.2.11	DMA_SrcProtect	383
7.9.2.12	DMA_TransfersTotal	383
7.10	Структура DMA_ConfigData_TypeDef	383
7.10.1	Подробное описание	383
7.10.2	Поля	384
7.10.2.1	ALT_DATA	384
7.10.2.2	PRM_DATA	384
7.10.2.3	RESERVED0	384
7.10.2.4	RESERVED1	384

7.11 Структура DMA_ConfigStruct_TypeDef	384
7.11.1 Подробное описание	384
7.11.2 Поля	384
7.11.2.1 CH	384
7.12 Структура DMA_Init_TypeDef	385
7.12.1 Подробное описание	385
7.12.2 Поля	385
7.12.2.1 DMA_Channel	385
7.12.2.2 DMA_ChannelEnable	385
7.12.2.3 DMA_HighPriority	385
7.12.2.4 DMA_PrmAlt	385
7.12.2.5 DMA_Protection	386
7.12.2.6 DMA_ReqMask	386
7.12.2.7 DMA_UseBurst	386
7.13 Структура DMA_Protect_TypeDef	386
7.13.1 Подробное описание	386
7.13.2 Поля	386
7.13.2.1 BUFFERABLE	386
7.13.2.2 CACHEABLE	387
7.13.2.3 PRIVELGED	387
7.14 Структура EXTMEM_Init_TypeDef	387
7.14.1 Подробное описание	387
7.14.2 Поля	387
7.14.2.1 CEMask	387
7.14.2.2 EXTMEM_ReadWaitState	387
7.14.2.3 EXTMEM_RWWaitState	388
7.14.2.4 EXTMEM_Width	388
7.14.2.5 EXTMEM_WriteWaitState	388
7.15 Структура GPIO_Init_TypeDef	388
7.15.1 Подробное описание	388
7.15.2 Поля	389
7.15.2.1 GPIO_AltFunc	389
7.15.2.2 GPIO_Dir	389
7.15.2.3 GPIO_Load	389
7.15.2.4 GPIO_Mode	389
7.15.2.5 GPIO_Out	389
7.15.2.6 GPIO_OutMode	389
7.15.2.7 GPIO_Pin	389
7.15.2.8 GPIO_PullUp	390
7.16 Структура RCC_PLLInit_TypeDef	390

7.16.1	Подробное описание	390
7.16.2	Поля	390
7.16.2.1	RCC_PLLDiv	390
7.16.2.2	RCC_PLLNF	390
7.16.2.3	RCC_PLLNO	391
7.16.2.4	RCC_PLLNR	391
7.16.2.5	RCC_PLLRef	391
7.17	Структура RTC_Date_TypeDef	391
7.17.1	Подробное описание	391
7.17.2	Поля	391
7.17.2.1	RTC_Day	391
7.17.2.2	RTC_Month	392
7.17.2.3	RTC_Weekday	392
7.17.2.4	RTC_Year	392
7.18	Структура RTC_Time_TypeDef	392
7.18.1	Подробное описание	392
7.18.2	Поля	392
7.18.2.1	RTC_Hour	392
7.18.2.2	RTC_Minute	393
7.18.2.3	RTC_Psecond	393
7.18.2.4	RTC_Second	393
7.19	Структура UART_Init_TypeDef	393
7.19.1	Подробное описание	393
7.19.2	Поля	393
7.19.2.1	UART_BaudRate	393
7.19.2.2	UART_ClkFreq	394
7.19.2.3	UART_DataWidth	394
7.19.2.4	UART_FIFOEn	394
7.19.2.5	UART_FIFOLevelRx	394
7.19.2.6	UART_FIFOLevelTx	394
7.19.2.7	UART_ParityBit	394
7.19.2.8	UART_RxEn	394
7.19.2.9	UART_StopBit	395
7.19.2.10	UART_TxEn	395
7.20	Структура UART_ModemInit_TypeDef	395
7.20.1	Подробное описание	395
7.20.2	Поля	395
7.20.2.1	UART_CTSEn	395
7.20.2.2	UART_InvDTR	396
7.20.2.3	UART_InvRTS	396

7.20.2.4	UART_RTSEn	396
8	Файлы	397
8.1	Файл niietcm4.h	397
8.1.1	Подробное описание	398
8.1.2	Макросы	399
8.1.2.1	EXT_OSC_VALUE	399
8.1.2.2	INT_OSC_VALUE	399
8.1.2.3	IS_CAP_ALL_PERIPH	399
8.1.2.4	IS_GPIO_ALL_PERIPH	399
8.1.2.5	IS_SPI_ALL_PERIPH	400
8.1.2.6	IS_TIMER_ALL_PERIPH	400
8.1.2.7	IS_UART_ALL_PERIPH	400
8.2	Файл niietcm4_adc.c	401
8.2.1	Подробное описание	403
8.2.2	Функции	403
8.2.2.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	403
8.2.2.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	404
8.2.2.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module)	404
8.2.2.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_Module)	404
8.2.2.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	405
8.2.2.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	405
8.2.2.7	ADC_DC_ITConfig(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef ADC_DC_Condition)	405
8.2.2.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	406
8.2.2.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	406
8.2.2.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	406
8.2.2.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	407
8.2.2.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_Module)	407
8.2.2.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	407
8.2.2.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	408
8.2.2.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC_Module)	408

8.2.2.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	408
8.2.2.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *ADC_InitStruct)	409
8.2.2.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	410
8.2.2.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	410
8.2.2.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	410
8.2.2.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)	411
8.2.2.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	411
8.2.2.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	411
8.2.2.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	412
8.2.2.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	412
8.2.2.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	412
8.2.2.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	413
8.2.2.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	413
8.2.2.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	413
8.2.2.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	414
8.2.2.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	415
8.2.2.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	415
8.2.2.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	415
8.2.2.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	416
8.2.2.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	416
8.2.2.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	416
8.2.2.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	417
8.2.2.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	417
8.2.2.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	417

8.2.2.40	ADC_SEQ_SWReq()	418
8.2.2.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	418
8.3	Файл niietcm4_adc.h	418
8.3.1	Подробное описание	424
8.3.2	Макросы	425
8.3.2.1	ADC_Channel_0	425
8.3.2.2	ADC_Channel_1	425
8.3.2.3	ADC_Channel_10	425
8.3.2.4	ADC_Channel_11	425
8.3.2.5	ADC_Channel_12	425
8.3.2.6	ADC_Channel_13	425
8.3.2.7	ADC_Channel_14	425
8.3.2.8	ADC_Channel_15	425
8.3.2.9	ADC_Channel_16	426
8.3.2.10	ADC_Channel_17	426
8.3.2.11	ADC_Channel_18	426
8.3.2.12	ADC_Channel_19	426
8.3.2.13	ADC_Channel_2	426
8.3.2.14	ADC_Channel_20	426
8.3.2.15	ADC_Channel_21	426
8.3.2.16	ADC_Channel_22	426
8.3.2.17	ADC_Channel_23	426
8.3.2.18	ADC_Channel_3	426
8.3.2.19	ADC_Channel_4	427
8.3.2.20	ADC_Channel_5	427
8.3.2.21	ADC_Channel_6	427
8.3.2.22	ADC_Channel_7	427
8.3.2.23	ADC_Channel_8	427
8.3.2.24	ADC_Channel_9	427
8.3.2.25	ADC_Channel_All	427
8.3.2.26	ADC_Channel_None	427
8.3.2.27	ADC_DC_0	427
8.3.2.28	ADC_DC_1	428
8.3.2.29	ADC_DC_10	428
8.3.2.30	ADC_DC_11	428
8.3.2.31	ADC_DC_12	428
8.3.2.32	ADC_DC_13	428
8.3.2.33	ADC_DC_14	428
8.3.2.34	ADC_DC_15	428
8.3.2.35	ADC_DC_16	428

8.3.2.36	ADC_DC_17	428
8.3.2.37	ADC_DC_18	428
8.3.2.38	ADC_DC_19	429
8.3.2.39	ADC_DC_2	429
8.3.2.40	ADC_DC_20	429
8.3.2.41	ADC_DC_21	429
8.3.2.42	ADC_DC_22	429
8.3.2.43	ADC_DC_23	429
8.3.2.44	ADC_DC_3	429
8.3.2.45	ADC_DC_4	429
8.3.2.46	ADC_DC_5	429
8.3.2.47	ADC_DC_6	430
8.3.2.48	ADC_DC_7	430
8.3.2.49	ADC_DC_8	430
8.3.2.50	ADC_DC_9	430
8.3.2.51	ADC_DC_All	430
8.3.2.52	ADC_DC_None	430
8.3.2.53	ADC_SEQ_0	430
8.3.2.54	ADC_SEQ_1	430
8.3.2.55	ADC_SEQ_2	430
8.3.2.56	ADC_SEQ_3	431
8.3.2.57	ADC_SEQ_4	431
8.3.2.58	ADC_SEQ_5	431
8.3.2.59	ADC_SEQ_6	431
8.3.2.60	ADC_SEQ_7	431
8.3.2.61	IS_ADC_AVERAGE	431
8.3.2.62	IS_ADC_DC_CHANNEL	431
8.3.2.63	IS_ADC_DC_CONDITION	432
8.3.2.64	IS_ADC_DC_MODE	432
8.3.2.65	IS_ADC_DC_MODULE	432
8.3.2.66	IS_ADC_MEASURE	433
8.3.2.67	IS_ADC_MODE	433
8.3.2.68	IS_ADC_MODULE	433
8.3.2.69	IS_ADC_RESOLUTION	434
8.3.2.70	IS_ADC_SEQ_FIFO_LEVEL	434
8.3.2.71	IS_ADC_SEQ_MODULE	434
8.3.2.72	IS_ADC_SEQ_START_EVENT	434
8.3.3	Перечисления	435
8.3.3.1	ADC_Average_TypeDef	435
8.3.3.2	ADC_DC_Channel_TypeDef	435

8.3.3.3	ADC_DC_Condition_TypeDef	436
8.3.3.4	ADC_DC_Mode_TypeDef	436
8.3.3.5	ADC_DC_Module_TypeDef	437
8.3.3.6	ADC_Measure_TypeDef	437
8.3.3.7	ADC_Mode_TypeDef	437
8.3.3.8	ADC_Module_TypeDef	438
8.3.3.9	ADC_Resolution_TypeDef	438
8.3.3.10	ADC_SEQ_FIFOLevel_TypeDef	438
8.3.3.11	ADC_SEQ_Module_TypeDef	439
8.3.3.12	ADC_SEQ_StartEvent_TypeDef	439
8.3.4	Функции	439
8.3.4.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	439
8.3.4.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	440
8.3.4.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module)	440
8.3.4.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	440
8.3.4.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↔ DC_DC_Init_TypeDef *ADC_DC_InitStruct)	441
8.3.4.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	442
8.3.4.7	ADC_DC_ITConfig(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition↔ _TypeDef ADC_DC_Condition)	442
8.3.4.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	443
8.3.4.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	444
8.3.4.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	444
8.3.4.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	444
8.3.4.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	445
8.3.4.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	445
8.3.4.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	445
8.3.4.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	446
8.3.4.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	446
8.3.4.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↔ Def *ADC_InitStruct)	446
8.3.4.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	447

8.3.4.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	447
8.3.4.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State)	447
8.3.4.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ _Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)	447
8.3.4.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↔ SEQ_Module)	448
8.3.4.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef A↔ DC_SEQ_Module)	448
8.3.4.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC↔ _SEQ_Module)	448
8.3.4.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	449
8.3.4.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_S↔ EQ_Module)	449
8.3.4.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef AD↔ C_SEQ_Module)	449
8.3.4.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef AD↔ C_SEQ_Module)	450
8.3.4.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SE↔ Q_Module)	451
8.3.4.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SE↔ Q_Module)	451
8.3.4.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ↔ _Module)	451
8.3.4.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	452
8.3.4.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State)	452
8.3.4.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_↔ ITCountSEQRst)	452
8.3.4.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ↔ _Module)	453
8.3.4.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_S↔ EQ_Module)	453
8.3.4.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ↔ _Module)	453
8.3.4.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SE↔ Q_Module)	454
8.3.4.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_Init↔ Struct)	454
8.3.4.40	ADC_SEQ_SWReq()	454
8.3.4.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	455
8.4	Файл niietcm4_bootflash.c	456
8.4.1	Подробное описание	456

8.4.2	Функции	457
8.4.2.1	BOOTFLASH_FullErase()	457
8.4.2.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	457
8.4.2.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	457
8.4.2.4	BOOTFLASH_Init(uint32_t SysClkFreq)	458
8.4.2.5	BOOTFLASH_ITCmd(FunctionalState State)	458
8.4.2.6	BOOTFLASH_OperationStatus()	458
8.4.2.7	BOOTFLASH_OperationStatusClear()	458
8.4.2.8	BOOTFLASH_PageErase(uint32_t PageNum)	459
8.4.2.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	459
8.5	Файл niietcm4_bootflash.h	459
8.5.1	Подробное описание	460
8.5.2	Макросы	461
8.5.2.1	BOOTFLASH_INFO_PAGE_SIZE_BYTES	461
8.5.2.2	BOOTFLASH_INFO_PAGE_TOTAL	461
8.5.2.3	BOOTFLASH_INFO_TOTAL_BYTES	461
8.5.2.4	BOOTFLASH_PAGE_SIZE_BYTES	461
8.5.2.5	BOOTFLASH_PAGE_TOTAL	462
8.5.2.6	BOOTFLASH_TOTAL_BYTES	462
8.5.2.7	IS_BOOTFLASH_STATUS	462
8.5.3	Перечисления	462
8.5.3.1	BOOTFLASH_Status_TypeDef	462
8.5.4	Функции	462
8.5.4.1	BOOTFLASH_FullErase()	462
8.5.4.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	462
8.5.4.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	463
8.5.4.4	BOOTFLASH_Init(uint32_t SysClkFreq)	463
8.5.4.5	BOOTFLASH_ITCmd(FunctionalState State)	463
8.5.4.6	BOOTFLASH_OperationStatus()	464
8.5.4.7	BOOTFLASH_OperationStatusClear()	464
8.5.4.8	BOOTFLASH_PageErase(uint32_t PageNum)	464
8.5.4.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	464
8.6	Файл niietcm4_cap.c	464
8.6.1	Подробное описание	466
8.6.2	Функции	467
8.6.2.1	CAP_Capture_Cmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	467
8.6.2.2	CAP_Capture_GetCap0(NT_CAP_TypeDef *CAPx)	467

8.6.2.3	CAP_Capture_GetCap1(NT_CAP_TypeDef *CAPx)	467
8.6.2.4	CAP_Capture_GetCap2(NT_CAP_TypeDef *CAPx)	468
8.6.2.5	CAP_Capture_GetCap3(NT_CAP_TypeDef *CAPx)	468
8.6.2.6	CAP_Capture_Init(NT_CAP_TypeDef *CAPx, CAP_Capture_Init↵ _TypeDef *CAP_Capture_InitStruct)	468
8.6.2.7	CAP_Capture_SetCap0(NT_CAP_TypeDef *CAPx, uint32_t Value) .	469
8.6.2.8	CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value) .	469
8.6.2.9	CAP_Capture_SetCap2(NT_CAP_TypeDef *CAPx, uint32_t Value) .	469
8.6.2.10	CAP_Capture_SetCap3(NT_CAP_TypeDef *CAPx, uint32_t Value) .	469
8.6.2.11	CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_↵ Capture_InitStruct)	470
8.6.2.12	CAP_DeInit(NT_CAP_TypeDef *CAPx)	470
8.6.2.13	CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx)	470
8.6.2.14	CAP_GetTimer(NT_CAP_TypeDef *CAPx)	471
8.6.2.15	CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_↵ InitStruct)	471
8.6.2.16	CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)	471
8.6.2.17	CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_IT↵ Source)	472
8.6.2.18	CAP_ITPendClear(NT_CAP_TypeDef *CAPx)	472
8.6.2.19	CAP_ITPendStatus(NT_CAP_TypeDef *CAPx)	472
8.6.2.20	CAP_ITStatus(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource) .	472
8.6.2.21	CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_IT↵ Source)	473
8.6.2.22	CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx)	473
8.6.2.23	CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx)	473
8.6.2.24	CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx)	474
8.6.2.25	CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx)	475
8.6.2.26	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_↵ TypeDef *CAP_PWM_InitStruct)	475
8.6.2.27	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_t ↵ t CompareVal)	475
8.6.2.28	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal) .	476
8.6.2.29	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_t ↵ _t CompareVal)	476
8.6.2.30	CAP_PWM_SetShadowPeriod(NT_CAP_TypeDef *CAPx, uint32_t ↵ t PeriodVal)	476
8.6.2.31	CAP_PWM_StructInit(CAP_PWM_Init_TypeDef *CAP_PWM_↵ InitStruct)	477
8.6.2.32	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal) .	478
8.6.2.33	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	478
8.6.2.34	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct)	478

8.6.2.35	CAP_SwSync(NT_CAP_TypeDef *CAPx)	479
8.6.2.36	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	479
8.6.2.37	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	479
8.7	Файл niectm4_cap.h	479
8.7.1	Подробное описание	483
8.7.2	Макросы	483
8.7.2.1	CAP_ITSource_All	483
8.7.2.2	CAP_ITSource_CapEvent0	483
8.7.2.3	CAP_ITSource_CapEvent1	483
8.7.2.4	CAP_ITSource_CapEvent2	483
8.7.2.5	CAP_ITSource_CapEvent3	484
8.7.2.6	CAP_ITSource_GeneralInt	484
8.7.2.7	CAP_ITSource_TimerEqCompare	484
8.7.2.8	CAP_ITSource_TimerEqPeriod	484
8.7.2.9	CAP_ITSource_TimerOvf	484
8.7.2.10	IS_CAP_CAPTURE_MODE	484
8.7.2.11	IS_CAP_CAPTURE_POLARITY	484
8.7.2.12	IS_CAP_HALT	485
8.7.2.13	IS_CAP_IT_SOURCE_SINGLE	485
8.7.2.14	IS_CAP_MODE	485
8.7.2.15	IS_CAP_PWM_POLARITY	485
8.7.2.16	IS_CAP_SYNC_OUT	486
8.7.3	Перечисления	486
8.7.3.1	CAP_Capture_Mode_TypeDef	486
8.7.3.2	CAP_Capture_Polarity_TypeDef	486
8.7.3.3	CAP_Halt_TypeDef	486
8.7.3.4	CAP_Mode_TypeDef	486
8.7.3.5	CAP_PWM_Polarity_TypeDef	487
8.7.3.6	CAP_SyncOut_TypeDef	487
8.7.4	Функции	487
8.7.4.1	CAP_Capture_Cmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	487
8.7.4.2	CAP_Capture_GetCap0(NT_CAP_TypeDef *CAPx)	487
8.7.4.3	CAP_Capture_GetCap1(NT_CAP_TypeDef *CAPx)	488
8.7.4.4	CAP_Capture_GetCap2(NT_CAP_TypeDef *CAPx)	488
8.7.4.5	CAP_Capture_GetCap3(NT_CAP_TypeDef *CAPx)	488
8.7.4.6	CAP_Capture_Init(NT_CAP_TypeDef *CAPx, CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	489
8.7.4.7	CAP_Capture_SetCap0(NT_CAP_TypeDef *CAPx, uint32_t Value)	490
8.7.4.8	CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value)	490
8.7.4.9	CAP_Capture_SetCap2(NT_CAP_TypeDef *CAPx, uint32_t Value)	490

8.7.4.10	CAP_Capture_SetCap3(NT_CAP_TypeDef *CAPx, uint32_t Value)	491
8.7.4.11	CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	491
8.7.4.12	CAP_DeInit(NT_CAP_TypeDef *CAPx)	491
8.7.4.13	CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx)	492
8.7.4.14	CAP_GetTimer(NT_CAP_TypeDef *CAPx)	492
8.7.4.15	CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_InitStruct)	492
8.7.4.16	CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)	493
8.7.4.17	CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	493
8.7.4.18	CAP_ITPendClear(NT_CAP_TypeDef *CAPx)	493
8.7.4.19	CAP_ITPendStatus(NT_CAP_TypeDef *CAPx)	493
8.7.4.20	CAP_ITStatus(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	494
8.7.4.21	CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	494
8.7.4.22	CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx)	494
8.7.4.23	CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx)	495
8.7.4.24	CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx)	496
8.7.4.25	CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx)	496
8.7.4.26	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct)	496
8.7.4.27	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_t CompareVal)	497
8.7.4.28	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)	497
8.7.4.29	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_t CompareVal)	497
8.7.4.30	CAP_PWM_SetShadowPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)	497
8.7.4.31	CAP_PWM_StructInit(CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct)	498
8.7.4.32	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	498
8.7.4.33	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	498
8.7.4.34	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct)	499
8.7.4.35	CAP_SwSync(NT_CAP_TypeDef *CAPx)	499
8.7.4.36	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	499
8.7.4.37	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	499
8.8	Файл niietcm4_conf.h	500
8.8.1	Подробное описание	500
8.9	Файл niietcm4_dma.c	501
8.9.1	Подробное описание	502
8.9.2	Функции	502

8.9.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	502
8.9.2.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	503
8.9.2.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	503
8.9.2.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	503
8.9.2.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	504
8.9.2.6	DMA_ClearErrorStatus()	504
8.9.2.7	DMA_DeInit()	505
8.9.2.8	DMA_ErrorStatus()	506
8.9.2.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	506
8.9.2.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	506
8.9.2.11	DMA_MasterEnableCmd(FunctionalState State)	507
8.9.2.12	DMA_MasterEnableStatus()	507
8.9.2.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	507
8.9.2.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	507
8.9.2.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	508
8.9.2.16	DMA_StateStatus()	508
8.9.2.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	508
8.9.2.18	DMA_SWRequestCmd(uint32_t DMA_Channel)	509
8.9.2.19	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	509
8.9.2.20	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	509
8.10	Файл niietcm4_dma.h	510
8.10.1	Подробное описание	513
8.10.2	Макросы	514
8.10.2.1	CHANNEL_CFG_CYCLE_CTRL_Msk	514
8.10.2.2	CHANNEL_CFG_CYCLE_CTRL_Pos	514
8.10.2.3	CHANNEL_CFG_DST_INC_Msk	514
8.10.2.4	CHANNEL_CFG_DST_INC_Pos	514
8.10.2.5	CHANNEL_CFG_DST_PROT_CTRL_Msk	514
8.10.2.6	CHANNEL_CFG_DST_PROT_CTRL_Pos	514
8.10.2.7	CHANNEL_CFG_DST_SIZE_Msk	514
8.10.2.8	CHANNEL_CFG_DST_SIZE_Pos	514
8.10.2.9	CHANNEL_CFG_N_MINUS_1_Msk	515
8.10.2.10	CHANNEL_CFG_N_MINUS_1_Pos	515
8.10.2.11	CHANNEL_CFG_NEXT_USEBURST_Msk	515
8.10.2.12	CHANNEL_CFG_NEXT_USEBURST_Pos	515
8.10.2.13	CHANNEL_CFG_R_POWER_Msk	515
8.10.2.14	CHANNEL_CFG_R_POWER_Pos	515
8.10.2.15	CHANNEL_CFG_SRC_INC_Msk	515

8.10.2.16 CHANNEL_CFG_SRC_INC_Pos	515
8.10.2.17 CHANNEL_CFG_SRC_PROT_CTRL_Msk	515
8.10.2.18 CHANNEL_CFG_SRC_PROT_CTRL_Pos	516
8.10.2.19 CHANNEL_CFG_SRC_SIZE_Msk	516
8.10.2.20 CHANNEL_CFG_SRC_SIZE_Pos	516
8.10.2.21 DMA_Channel_0	516
8.10.2.22 DMA_Channel_1	516
8.10.2.23 DMA_Channel_10	516
8.10.2.24 DMA_Channel_11	516
8.10.2.25 DMA_Channel_12	516
8.10.2.26 DMA_Channel_13	516
8.10.2.27 DMA_Channel_14	516
8.10.2.28 DMA_Channel_15	517
8.10.2.29 DMA_Channel_16	517
8.10.2.30 DMA_Channel_17	517
8.10.2.31 DMA_Channel_18	517
8.10.2.32 DMA_Channel_19	517
8.10.2.33 DMA_Channel_2	517
8.10.2.34 DMA_Channel_20	517
8.10.2.35 DMA_Channel_21	517
8.10.2.36 DMA_Channel_22	517
8.10.2.37 DMA_Channel_23	518
8.10.2.38 DMA_Channel_3	518
8.10.2.39 DMA_Channel_4	518
8.10.2.40 DMA_Channel_5	518
8.10.2.41 DMA_Channel_6	518
8.10.2.42 DMA_Channel_7	518
8.10.2.43 DMA_Channel_8	518
8.10.2.44 DMA_Channel_9	518
8.10.2.45 DMA_Channel_ADCSEQ0	518
8.10.2.46 DMA_Channel_ADCSEQ1	518
8.10.2.47 DMA_Channel_ADCSEQ2	519
8.10.2.48 DMA_Channel_ADCSEQ3	519
8.10.2.49 DMA_Channel_ADCSEQ4	519
8.10.2.50 DMA_Channel_ADCSEQ5	519
8.10.2.51 DMA_Channel_ADCSEQ6	519
8.10.2.52 DMA_Channel_ADCSEQ7	519
8.10.2.53 DMA_Channel_All	519
8.10.2.54 DMA_Channel_SPI0_RX	519
8.10.2.55 DMA_Channel_SPI0_TX	519

8.10.2.56 DMA_Channel_SPI1_RX	520
8.10.2.57 DMA_Channel_SPI1_TX	520
8.10.2.58 DMA_Channel_SPI2_RX	520
8.10.2.59 DMA_Channel_SPI2_TX	520
8.10.2.60 DMA_Channel_SPI3_RX	520
8.10.2.61 DMA_Channel_SPI3_TX	520
8.10.2.62 DMA_Channel_UART0_RX	520
8.10.2.63 DMA_Channel_UART0_TX	520
8.10.2.64 DMA_Channel_UART1_RX	520
8.10.2.65 DMA_Channel_UART1_TX	520
8.10.2.66 DMA_Channel_UART2_RX	521
8.10.2.67 DMA_Channel_UART2_TX	521
8.10.2.68 DMA_Channel_UART3_RX	521
8.10.2.69 DMA_Channel_UART3_TX	521
8.10.2.70 IS_DMA_ARBITRATION_RATE	521
8.10.2.71 IS_DMA_DATA_INC	521
8.10.2.72 IS_DMA_DATA_SIZE	522
8.10.2.73 IS_DMA_MODE	522
8.10.2.74 IS_DMA_STATE	522
8.10.2.75 IS_GET_DMA_CHANNEL	522
8.10.3 Перечисления	523
8.10.3.1 DMA_ArbitrationRate_TypeDef	523
8.10.3.2 DMA_DataInc_TypeDef	524
8.10.3.3 DMA_DataSize_TypeDef	524
8.10.3.4 DMA_Mode_TypeDef	524
8.10.3.5 DMA_State_TypeDef	524
8.10.4 Функции	525
8.10.4.1 DMA_BasePtrConfig(uint32_t BasePtr)	525
8.10.4.2 DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	525
8.10.4.3 DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	525
8.10.4.4 DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	526
8.10.4.5 DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	526
8.10.4.6 DMA_ClearErrorStatus()	527
8.10.4.7 DMA_DeInit()	527
8.10.4.8 DMA_ErrorStatus()	527
8.10.4.9 DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	527
8.10.4.10 DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	528
8.10.4.11 DMA_MasterEnableCmd(FunctionalState State)	528

8.10.4.12 DMA_MasterEnableStatus()	528
8.10.4.13 DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	529
8.10.4.14 DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	529
8.10.4.15 DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	529
8.10.4.16 DMA_StateStatus()	530
8.10.4.17 DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	530
8.10.4.18 DMA_SWRequestCmd(uint32_t DMA_Channel)	530
8.10.4.19 DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	530
8.10.4.20 DMA_WaitOnReqStatus(uint32_t DMA_Channel)	531
8.11 Файл niietcm4_extmem.c	531
8.11.1 Подробное описание	531
8.11.2 Макросы	532
8.11.2.1 EXT_MEM_CFG_Reset_Value	532
8.11.3 Функции	532
8.11.3.1 EXTMEM_DeInit()	532
8.11.3.2 EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	532
8.11.3.3 EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	533
8.12 Файл niietcm4_extmem.h	533
8.12.1 Подробное описание	534
8.12.2 Макросы	535
8.12.2.1 EXTMEM_CEMask_Addr_11	535
8.12.2.2 EXTMEM_CEMask_Addr_11_19	535
8.12.2.3 EXTMEM_CEMask_Addr_12	535
8.12.2.4 EXTMEM_CEMask_Addr_13	535
8.12.2.5 EXTMEM_CEMask_Addr_14	535
8.12.2.6 EXTMEM_CEMask_Addr_15	536
8.12.2.7 EXTMEM_CEMask_Addr_16	536
8.12.2.8 EXTMEM_CEMask_Addr_17	536
8.12.2.9 EXTMEM_CEMask_Addr_18	536
8.12.2.10 EXTMEM_CEMask_Addr_19	536
8.12.2.11 IS_EXTMEM_READ_WAITSTATE	536
8.12.2.12 IS_EXTMEM_RW_WAITSTATE	536
8.12.2.13 IS_EXTMEM_WIDTH	537
8.12.2.14 IS_EXTMEM_WRITE_WAITSTATE	537
8.12.3 Перечисления	537
8.12.3.1 EXTMEM_ReadWaitState_TypeDef	537
8.12.3.2 EXTMEM_RWWaitState_TypeDef	538
8.12.3.3 EXTMEM_Width_TypeDef	538
8.12.3.4 EXTMEM_WriteWaitState_TypeDef	538
8.12.4 Функции	538

8.12.4.1	EXTMEM_DeInit()	538
8.12.4.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	539
8.12.4.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	539
8.13	Файл n1ietcm4_gpio.c	539
8.13.1	Подробное описание	541
8.13.2	Макросы	541
8.13.2.1	GPIO_DATAOUT_Reset_Value	541
8.13.2.2	GPIO_GPIODEN0_Reset_Value	542
8.13.2.3	GPIO_GPIODEN1_Reset_Value	542
8.13.2.4	GPIO_GPIODEN2_Reset_Value	542
8.13.2.5	GPIO_GPIODEN3_Reset_Value	542
8.13.2.6	GPIO_GPIOODCTLx_Reset_Value	542
8.13.2.7	GPIO_GPIOODSCTLx_Reset_Value	542
8.13.2.8	GPIO_GPIOPCTLx_Reset_Value	542
8.13.2.9	GPIO_GPIOPUCTLx_Reset_Value	542
8.13.2.10	GPIO_GPIOQEx_Reset_Value	543
8.13.2.11	GPIO_GPIOQMx_Reset_Value	543
8.13.2.12	GPIO_GPIOQPx_Reset_Value	543
8.13.2.13	GPIO_GPIOSEx_Reset_Value	543
8.13.2.14	GPIO_Regs_A_C_E_G_Mask	543
8.13.2.15	GPIO_Regs_B_D_F_H_Mask	543
8.13.2.16	GPIO_Regs_GPIOA_Mask	543
8.13.2.17	GPIO_Regs_GPIOB_Mask	543
8.13.2.18	GPIO_Regs_GPIOC_Mask	544
8.13.2.19	GPIO_Regs_GPIOD_Mask	544
8.13.2.20	GPIO_Regs_GPIOE_Mask	544
8.13.2.21	GPIO_Regs_GPIOF_Mask	544
8.13.2.22	GPIO_Regs_GPIOG_Mask	544
8.13.2.23	GPIO_Regs_GPIOH_Mask	544
8.13.3	Функции	544
8.13.3.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	544
8.13.3.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	544
8.13.3.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	545
8.13.3.4	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	545
8.13.3.5	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	546
8.13.3.6	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	546

8.13.3.7	<code>GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)</code>	546
8.13.3.8	<code>GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)</code>	547
8.13.3.9	<code>GPIO_Read(NT_GPIO_TypeDef *GPIOx)</code>	548
8.13.3.10	<code>GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)</code>	548
8.13.3.11	<code>GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)</code>	548
8.13.3.12	<code>GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)</code>	549
8.13.3.13	<code>GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)</code>	549
8.13.3.14	<code>GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)</code>	549
8.13.3.15	<code>GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)</code>	550
8.13.3.16	<code>GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)</code>	550
8.13.3.17	<code>GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)</code>	550
8.13.3.18	<code>GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)</code>	551
8.14	Файл <code>niietcm4_gpio.h</code>	551
8.14.1	Подробное описание	554
8.14.2	Макросы	554
8.14.2.1	<code>GPIO_Pin_0</code>	554
8.14.2.2	<code>GPIO_Pin_0_3</code>	554
8.14.2.3	<code>GPIO_Pin_0_7</code>	555
8.14.2.4	<code>GPIO_Pin_1</code>	555
8.14.2.5	<code>GPIO_Pin_10</code>	555
8.14.2.6	<code>GPIO_Pin_11</code>	555
8.14.2.7	<code>GPIO_Pin_12</code>	555
8.14.2.8	<code>GPIO_Pin_12_15</code>	555
8.14.2.9	<code>GPIO_Pin_13</code>	555
8.14.2.10	<code>GPIO_Pin_14</code>	555
8.14.2.11	<code>GPIO_Pin_15</code>	555
8.14.2.12	<code>GPIO_Pin_2</code>	555
8.14.2.13	<code>GPIO_Pin_3</code>	556
8.14.2.14	<code>GPIO_Pin_4</code>	556
8.14.2.15	<code>GPIO_Pin_4_7</code>	556
8.14.2.16	<code>GPIO_Pin_5</code>	556
8.14.2.17	<code>GPIO_Pin_6</code>	556
8.14.2.18	<code>GPIO_Pin_7</code>	556
8.14.2.19	<code>GPIO_Pin_8</code>	556
8.14.2.20	<code>GPIO_Pin_8_11</code>	556
8.14.2.21	<code>GPIO_Pin_8_15</code>	556

8.14.2.22	GPIO_Pin_9	557
8.14.2.23	GPIO_Pin_All	557
8.14.2.24	IS_GET_GPIO_PIN	557
8.14.2.25	IS_GPIO_ALT_FUNC	557
8.14.2.26	IS_GPIO_DIR	557
8.14.2.27	IS_GPIO_INT_POL	558
8.14.2.28	IS_GPIO_INT_TYPE	558
8.14.2.29	IS_GPIO_LOAD	558
8.14.2.30	IS_GPIO_MODE	558
8.14.2.31	IS_GPIO_OUT	558
8.14.2.32	IS_GPIO_OUT_MODE	559
8.14.2.33	IS_GPIO_PULLUP	559
8.14.2.34	IS_GPIO_QUAL	559
8.14.2.35	IS_GPIO_QUAL_MODE	559
8.14.2.36	IS_GPIO_SYNC	559
8.14.3	Перечисления	560
8.14.3.1	BitAction	560
8.14.3.2	GPIO_AltFunc_TypeDef	560
8.14.3.3	GPIO_Dir_TypeDef	560
8.14.3.4	GPIO_IntPol_TypeDef	560
8.14.3.5	GPIO_IntType_TypeDef	560
8.14.3.6	GPIO_Load_TypeDef	561
8.14.3.7	GPIO_Mode_TypeDef	561
8.14.3.8	GPIO_Out_TypeDef	561
8.14.3.9	GPIO_OutMode_TypeDef	561
8.14.3.10	GPIO_PullUp_TypeDef	561
8.14.3.11	GPIO_Qual_TypeDef	562
8.14.3.12	GPIO_QualMode_TypeDef	562
8.14.3.13	GPIO_Sync_TypeDef	562
8.14.4	Функции	562
8.14.4.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	562
8.14.4.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	562
8.14.4.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	563
8.14.4.4	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	563
8.14.4.5	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	564
8.14.4.6	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	564

8.14.4.7	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	564
8.14.4.8	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	565
8.14.4.9	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	566
8.14.4.10	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	566
8.14.4.11	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	566
8.14.4.12	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	567
8.14.4.13	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	567
8.14.4.14	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	567
8.14.4.15	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	568
8.14.4.16	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	568
8.14.4.17	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	568
8.14.4.18	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	569
8.15	Файл niietcm4_rcc.c	569
8.15.1	Подробное описание	570
8.15.2	Макросы	571
8.15.2.1	RCC_PLL_CTRL_Reset_Value	571
8.15.2.2	RCC_PLL_NF_Reset_Value	571
8.15.2.3	RCC_PLL_NR_Reset_Value	571
8.15.2.4	RCC_PLL_OD_Reset_Value	571
8.15.3	Функции	571
8.15.3.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	571
8.15.3.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	572
8.15.3.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	572
8.15.3.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	573
8.15.3.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	573
8.15.3.6	RCC_PLLEInit()	574
8.15.3.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	574
8.15.3.8	RCC_PLLPowerDownCmd(FunctionalState State)	575
8.15.3.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	575
8.15.3.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPISx, FunctionalState State)	575
8.15.3.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPISx, uint32_t DivVal, FunctionalState DivState)	576
8.15.3.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPISx, RCC_SPIClk_TypeDef RCC_SPIClk)	576

8.15.3.13	RCC_SysClkDiv2Out(FunctionalState State)	576
8.15.3.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	577
8.15.3.15	RCC_SysClkStatus()	577
8.15.3.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	577
8.15.3.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	578
8.15.3.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	578
8.15.3.19	RCC_USBClkCmd(FunctionalState State)	578
8.15.3.20	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	579
8.15.3.21	RCC_WaitClkChange(RCC_SysClk_TypeDef RCC_SysClk)	579
8.16	Файл niyetcm4_rcc.h	579
8.16.1	Подробное описание	582
8.16.2	Макросы	583
8.16.2.1	IS_RCC_ADC_CLK	583
8.16.2.2	IS_RCC_PERIPH_CLK	583
8.16.2.3	IS_RCC_PLL_NO	584
8.16.2.4	IS_RCC_PLL_REF	584
8.16.2.5	IS_RCC_SPI_CLK	584
8.16.2.6	IS_RCC_SYS_CLK	584
8.16.2.7	IS_RCC_UART_CLK	585
8.16.2.8	IS_RCC_USB_CLK	585
8.16.2.9	IS_RCC_USB_FREQ	585
8.16.2.10	RCC_CLK_CHANGE_TIMEOUT	585
8.16.2.11	RCC_CLK_PLL_STABLE_TIMEOUT	585
8.16.3	Перечисления	585
8.16.3.1	RCC_ADCClk_TypeDef	585
8.16.3.2	RCC_PeriphClk_TypeDef	586
8.16.3.3	RCC_PeriphRst_TypeDef	586
8.16.3.4	RCC_PLLNO_TypeDef	587
8.16.3.5	RCC_PLLRef_TypeDef	588
8.16.3.6	RCC_SPIClk_TypeDef	588
8.16.3.7	RCC_SysClk_TypeDef	588
8.16.3.8	RCC_UARTClk_TypeDef	588
8.16.3.9	RCC_USBClk_TypeDef	589
8.16.3.10	RCC_USBFreq_TypeDef	589
8.16.4	Функции	589
8.16.4.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	589

8.16.4.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	589
8.16.4.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	590
8.16.4.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	590
8.16.4.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	591
8.16.4.6	RCC_PLLDeInit()	591
8.16.4.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	591
8.16.4.8	RCC_PLLPowerDownCmd(FunctionalState State)	592
8.16.4.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	593
8.16.4.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	593
8.16.4.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	593
8.16.4.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	594
8.16.4.13	RCC_SysClkDiv2Out(FunctionalState State)	594
8.16.4.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	594
8.16.4.15	RCC_SysClkStatus()	595
8.16.4.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	595
8.16.4.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	595
8.16.4.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	596
8.16.4.19	RCC_USBClkCmd(FunctionalState State)	596
8.16.4.20	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	596
8.17	Файл niyetcm4_rtc.c	596
8.17.1	Подробное описание	597
8.18	Файл niyetcm4_rtc.h	597
8.18.1	Подробное описание	599
8.18.2	Макросы	599
8.18.2.1	IS_RTC_FORMAT	599
8.18.2.2	IS_RTC_MONTH	599
8.18.2.3	IS_RTC_WEEKDAY	600
8.18.3	Перечисления	600
8.18.3.1	RTC_Format_TypeDef	600
8.18.3.2	RTC_Month_TypeDef	600
8.18.3.3	RTC_Weekday_TypeDef	601
8.19	Файл niyetcm4_timer.c	601
8.19.1	Подробное описание	602

8.19.2	Функции	602
8.19.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	602
8.19.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	602
8.19.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	603
8.19.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	603
8.19.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	603
8.19.2.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	604
8.19.2.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	604
8.19.2.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	604
8.19.2.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	605
8.19.2.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	605
8.19.2.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	605
8.20	Файл niyetcm4_timer.h	605
8.20.1	Подробное описание	606
8.20.2	Макросы	607
8.20.2.1	IS_TIMER_EXT_INPUT	607
8.20.3	Перечисления	607
8.20.3.1	TIMER_ExtInput_TypeDef	607
8.20.4	Функции	608
8.20.4.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	608
8.20.4.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	609
8.20.4.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	609
8.20.4.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	609
8.20.4.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	610
8.20.4.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	610
8.20.4.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	610
8.20.4.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	610
8.20.4.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	611
8.20.4.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	611
8.20.4.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	611
8.21	Файл niyetcm4_uart.c	612
8.21.1	Подробное описание	613

8.21.2	Функции	614
8.21.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	614
8.21.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	614
8.21.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	614
8.21.2.4	UART_DeInit(NT_UART_TypeDef *UARTx)	615
8.21.2.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	616
8.21.2.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	616
8.21.2.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error)	616
8.21.2.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	617
8.21.2.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag)	617
8.21.2.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	617
8.21.2.11	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)	618
8.21.2.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	618
8.21.2.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	618
8.21.2.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	619
8.21.2.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	619
8.21.2.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	619
8.21.2.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	620
8.21.2.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	620
8.21.2.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	620
8.21.2.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	620
8.22	Файл niyetcm4_uart.h	621
8.22.1	Подробное описание	624
8.22.2	Макросы	624
8.22.2.1	IS_UART_DATA_WIDTH	624
8.22.2.2	IS_UART_DIR	624
8.22.2.3	IS_UART_ERROR	625
8.22.2.4	IS_UART_FIFO_LEVEL	625
8.22.2.5	IS_UART_FLAG	625
8.22.2.6	IS_UART_GET_IT_SOURCE	625

8.22.2.7	IS_UART_PARITY_BIT	626
8.22.2.8	IS_UART_STOP_BIT	626
8.22.3	Перечисления	626
8.22.3.1	UART_DataWidth_TypeDef	626
8.22.3.2	UART_Dir_Typedef	627
8.22.3.3	UART_Error_Typedef	627
8.22.3.4	UART_FIFOLevel_TypeDef	627
8.22.3.5	UART_Flag_Typedef	627
8.22.3.6	UART_ITSource_Typedef	628
8.22.3.7	UART_ParityBit_TypeDef	628
8.22.3.8	UART_StopBit_TypeDef	628
8.22.4	Функции	628
8.22.4.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	628
8.22.4.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	629
8.22.4.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	629
8.22.4.4	UART_DeInit(NT_UART_TypeDef *UARTx)	629
8.22.4.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	630
8.22.4.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_Typedef UART_Dir, FunctionalState State)	630
8.22.4.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_Typedef UART_Error)	630
8.22.4.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	631
8.22.4.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_Typedef UART_Flag)	632
8.22.4.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	632
8.22.4.11	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_Typedef UART_ITSource, FunctionalState State)	632
8.22.4.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_Typedef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	633
8.22.4.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_ITSource_Typedef UART_ITSource)	633
8.22.4.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_ITSource_Typedef UART_ITSource)	633
8.22.4.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_ITSource_Typedef UART_ITSource)	634
8.22.4.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	634
8.22.4.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	634
8.22.4.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	635

8.22.4.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data) . . .	635
8.22.4.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	635
8.23	Файл niietcm4_userflash.c	636
8.23.1	Подробное описание	636
8.23.2	Функции	637
8.23.2.1	USERFLASH_FullErase()	637
8.23.2.2	USERFLASH_Info_PageErase(uint32_t PageNum)	637
8.23.2.3	USERFLASH_Info_Read(uint32_t Address)	637
8.23.2.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	638
8.23.2.5	USERFLASH_Init(uint32_t SysClkFreq)	638
8.23.2.6	USERFLASH_ITCmd(FunctionalState State)	638
8.23.2.7	USERFLASH_OperationStatus()	638
8.23.2.8	USERFLASH_OperationStatusClear()	639
8.23.2.9	USERFLASH_PageErase(uint32_t PageNum)	639
8.23.2.10	USERFLASH_Read(uint32_t Address)	639
8.23.2.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	639
8.24	Файл niietcm4_userflash.h	640
8.24.1	Подробное описание	641
8.24.2	Макросы	642
8.24.2.1	IS_USERFLASH_STATUS	642
8.24.2.2	USERFLASH_INFO_PAGE_SIZE_BYTES	642
8.24.2.3	USERFLASH_INFO_PAGE_TOTAL	642
8.24.2.4	USERFLASH_INFO_TOTAL_BYTES	642
8.24.2.5	USERFLASH_PAGE_SIZE_BYTES	642
8.24.2.6	USERFLASH_PAGE_TOTAL	642
8.24.2.7	USERFLASH_TOTAL_BYTES	642
8.24.3	Перечисления	643
8.24.3.1	USERFLASH_Status_TypeDef	643
8.24.4	Функции	643
8.24.4.1	USERFLASH_FullErase()	643
8.24.4.2	USERFLASH_Info_PageErase(uint32_t PageNum)	643
8.24.4.3	USERFLASH_Info_Read(uint32_t Address)	643
8.24.4.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	644
8.24.4.5	USERFLASH_Init(uint32_t SysClkFreq)	644
8.24.4.6	USERFLASH_ITCmd(FunctionalState State)	644
8.24.4.7	USERFLASH_OperationStatus()	644
8.24.4.8	USERFLASH_OperationStatusClear()	644
8.24.4.9	USERFLASH_PageErase(uint32_t PageNum)	645
8.24.4.10	USERFLASH_Read(uint32_t Address)	645
8.24.4.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	645

8.25	Файл niietcm4_watchdog.c	645
8.25.1	Подробное описание	646
8.25.2	Макросы	647
8.25.2.1	WATCHDOG_Lock_Value	647
8.25.2.2	WATCHDOG_Unlock_Value	647
8.25.3	Функции	647
8.25.3.1	WATCHDOG_Cmd(FunctionalState State)	647
8.25.3.2	WATCHDOG_GetCounter()	647
8.25.3.3	WATCHDOG_GetReload()	648
8.25.3.4	WATCHDOG_ITMaskedStatus()	648
8.25.3.5	WATCHDOG_ITRawStatus()	648
8.25.3.6	WATCHDOG_ITStatusClear()	648
8.25.3.7	WATCHDOG_LockCmd(FunctionalState State)	648
8.25.3.8	WATCHDOG_RstCmd(FunctionalState State)	649
8.25.3.9	WATCHDOG_SetReload(uint32_t ReloadVal)	649
8.26	Файл niietcm4_watchdog.h	649
8.26.1	Подробное описание	650
8.26.2	Функции	650
8.26.2.1	WATCHDOG_Cmd(FunctionalState State)	650
8.26.2.2	WATCHDOG_GetCounter()	651
8.26.2.3	WATCHDOG_GetReload()	651
8.26.2.4	WATCHDOG_ITMaskedStatus()	651
8.26.2.5	WATCHDOG_ITRawStatus()	651
8.26.2.6	WATCHDOG_ITStatusClear()	651
8.26.2.7	WATCHDOG_LockCmd(FunctionalState State)	652
8.26.2.8	WATCHDOG_RstCmd(FunctionalState State)	653
8.26.2.9	WATCHDOG_SetReload(uint32_t ReloadVal)	653
	Алфавитный указатель	655

Глава 1

Титульная страница

НИЕТСМ4 PD (Peripheral driver) - это комплект драйверов, предназначенных для ускорения и упрощения работы со внутренними периферийными устройствами микроконтроллеров на базе ядра ARM Cortex-M4 производства ОАО "НИИЭТ". Совместно с комплектом предоставляются:

- Шаблоны проектов для различных IDE.
- Примеры использования драйверов.
- Подробная документация.

На данный момент в проекте реализованна поддержка периферийных блоков следующих микроконтроллеров:

- K1921BK01T:
 - тактирование и сброс (RCC)
 - GPIO
 - DMA
 - UART
 - TIMER
 - RTC
 - BOOTFLASH
 - USERFLASH
 - EXTMEM
 - ADC
 - CAP
 - WATCHDOG

Загрузки

Текущую версию драйвера можно скачать по данной ссылке: [НИЕТСМ4 Peripheral Driver v0.8.0](#). Предыдущие версии могут быть найдены на странице [Загрузки](#). Историю релизов можно посмотреть [здесь](#).

Контакты

Разработка ведется полностью открыто - проект расположен в публичном репозитории на [BitBucket](#). Приветствуются пожелания, сообщения об ошибках, неточностях. Способы связи с разработчиками в порядке убывания предпочтительности:

- создание обсуждения в [репозитории](#);
- по электронной почте kolbov@niiet.ru;

Другие проекты, ориентированные на ARM Cortex-M4 микроконтроллеры НИИЭТ доступны на странице команды разработчиков на [BitBucket](#).

Глава 2

Информация о релизах

v0.8.0

Добавлено:

- драйвер для работы с блоками захвата (CAP)
- драйвер для работы со сторожевым таймером (WATCHDOG)

Исключено:

Исправлено:

- доработан алгоритм перехода на тактирование от PLL
- изменена функция управления сбросом периферии
- правка мелких неточностей

v0.7.0

Добавлено:

- драйвер для работы с АЦП (ADC)
- более подробное описание GPIO, BOOTFLASH, EXTMEM, DMA

Исключено:

Исправлено:

- мелкие неточности документации функций блока UART, DMA, RCC

v0.6.0

Добавлено:

- драйвер для работы с загрузочной флеш (BOOTFLASH)
- драйвер для работы с пользовательской флеш (USERFLASH)
- драйвер для работы с внешней памятью (EXTMEM)

Исключено:

Исправлено:

- мелкие неточности документации блока UART

v0.5.0

Добавлено:

- драйвер для управления блоками таймеров
- драйвер для RTC

Исключено:

Исправлено:

v0.4.0

Добавлено:

- драйвер для управления UART

Исключено:

Исправлено:

- баг невозможности инициализации нескольких пинов одного порта отдельно

v0.3.0

Добавлено:

- драйвер для управления DMA

Исключено:

Исправлено:

v0.2.0

Добавлено:

- драйвер для управления тактированием и сбросом (RCC) K1921BK01T

Исключено:

Исправлено:

- структура документации GPIO
- форматирование и структура драйвера GPIO

- некритичные изменения в общей архитектуре

v0.1.0

Добавлено:

- драйвер GPIO K1921BK01T
- темная и светлая темы документации
- генерация файла документации для QT .qch

Исключено:

Исправлено:

Глава 3

Алфавитный указатель групп

3.1 Группы

Полный список групп.

Драйвер периферии	362
Настройка драйвера	363
Макросы	364
Типы	365
Периферия	368
ADC	234
Константы	15
Маски каналов для измерений	16
Маски выбора цифровых компараторов	20
Маски выбора секвенсоров	24
Типы	26
Функции	37
Инициализация	43
Модули АЦП	44
Цифровые компараторы	46
Секвенсоры	48
Конфигурация секвенсоров для DMA	51
Конфигурация прерываний	54
Цифровые компараторы	55
Секвенсоры	59
Приватные данные	235
Приватные константы	236
Начальные значения регистров	237
Приватные функции	238
BOOTFLASH	255
Константы	62
Основная область флеш	63
Информационная область флеш	64
Типы	65
Функции	66
Основная область флеш	68
Информационная область флеш	70
Приватные данные	256
Приватные константы	257
Приватные функции	258
CAP	261
Типы	71

Константы	75
Маски источников прерываний	76
Функции	78
Конфигурация	79
Режим ШИМ	84
Режим захвата	89
Прерывания	94
Приватные данные	262
Приватные константы	263
Приватные функции	264
DMA	279
Константы	97
Маски для CHANNEL_CFG	98
Маски каналов DMA	101
Маски каналов по имени	103
Маски каналов по номеру	107
Типы	111
Функции	116
Инициализация каналов DMA	117
Инициализация контроллера DMA	119
Конфигурация контроллера DMA	121
Статусная информация	125
Приватные данные	280
Приватные константы	281
Начальные значения регистров	282
Приватные функции	283
EXTMEM	291
Константы	127
Маски адреса	128
Типы	130
Функции	134
Приватные данные	292
Приватные константы	293
Начальные значения регистров	294
Приватные функции	295
GPIO	297
Типы	136
Константы	143
Маски пинов	144
Функции	148
Инициализация и деинициализация	149
Чтение и запись	151
Чтение	152
Запись	154
Битовые операции	156
Фильтрация	158
Прерывания	160
Приватные данные	298
Приватные константы	299
Начальные значения регистров	300
Маски портов	303
Приватные функции	305
RCC	313
Константы	162
Типы	163
Функции	171

Конфигурация PLL	172
Управление тактированием	175
Тактирование USB	177
Тактирование UART	178
Тактирование SPI	181
Тактирование ADC	183
Управление сбросом	184
Приватные данные	314
Приватные константы	315
Начальные значения регистров	316
Приватные функции	317
RTC	326
Типы	185
Функции	188
Приватные данные	327
Приватные функции	328
TIMER	329
Типы	189
Константы	190
Функции	191
Конфигурация	192
Прерывания	196
Приватные данные	330
Приватные константы	331
Приватные функции	332
UART	336
Типы	198
Константы	204
Функции	205
Инициализация и деинициализация	207
Прием и передача	210
Режим модема	213
Прерывания	214
Настройка DMA	217
Приватные данные	337
Приватные константы	338
Приватные функции	339
USERFLASH	349
Константы	218
Основная область флеш	219
Информационная область флеш	220
Типы	221
Функции	222
Основная область флеш	224
Информационная область флеш	226
Приватные данные	350
Приватные константы	351
Приватные функции	352
WATCHDOG	356
Типы	228
Константы	229
Функции	230
Конфигурация	231
Прерывания	233
Приватные данные	357
Приватные константы	358
Приватные функции	359

Глава 4

Алфавитный указатель структур данных

4.1 Структуры данных

Структуры данных с их кратким описанием.

_CHANNEL_CFG_bits	Битовый доступ к регистру CHANNEL_CFG в DMA_Channel_TypeDef	369
ADC_DC_Init_TypeDef	Структура инициализации цифровых компараторов	371
ADC_Init_TypeDef	Структура инициализации модулей АЦП	373
ADC_SEQ_Init_TypeDef	Структура инициализации секвенсоров	374
CAP_Capture_Init_TypeDef	Структура инициализации режима захвата	375
CAP_Init_TypeDef	Структура инициализации блока захвата в целом	378
CAP_PWM_Init_TypeDef	Структура инициализации режима ШИМ	379
DMA_Channel_TypeDef	Тип, описывающий структуру канала DMA	380
DMA_ChannelInit_TypeDef	Структура инициализации канала DMA	381
DMA_ConfigData_TypeDef	Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ	383
DMA_ConfigStruct_TypeDef	Управляющая структура данных DMA	384
DMA_Init_TypeDef	Структура инициализации контроллера DMA	385
DMA_Protect_TypeDef	Защита шины при чтении из источника или записи в приемник через DMA	386
EXTMEM_Init_TypeDef	Структура инициализации внешней памяти	387
GPIO_Init_TypeDef	Структура инициализации GPIO	388
RCC_PLLInit_TypeDef	Структура инициализации PLL	390
RTC_Date_TypeDef	Структура даты	391
RTC_Time_TypeDef	Структура времени	392
UART_Init_TypeDef	Структура инициализации UART	393

UART_ModemInit_TypeDef	
Структура инициализации модемного режима	395

Глава 5

Список файлов

5.1 Файлы

Полный список документированных файлов.

niietcm4.h	Это главный заголовочный файл драйвера, обычно включаемый в main.c	397
niietcm4_adc.c	Файл содержит реализацию всех функции для работы с модулями АЦП, секвенсорами, цифровыми компараторами	401
niietcm4_adc.h	Файл содержит все прототипы функций для работы с АЦП, секвенсорами, цифровыми компараторами	418
niietcm4_bootflash.c	Файл содержит реализацию всех функции для работы с загрузочной флеш	456
niietcm4_bootflash.h	Файл содержит все прототипы функций для загрузочной флеш	459
niietcm4_cap.c	Файл содержит реализацию всех функции для работы с блоками захвата	464
niietcm4_cap.h	Файл содержит все прототипы функций для блоков захвата	479
niietcm4_conf.h	Файл конфигурации драйвера	500
niietcm4_dma.c	Файл содержит реализацию всех функции для работы с DMA	501
niietcm4_dma.h	Файл содержит все прототипы функций для DMA	510
niietcm4_extmem.c	Файл содержит реализацию всех функции для работы с интерфейсом внешней памяти	531
niietcm4_extmem.h	Файл содержит все прототипы функций для интерфейса внешней памяти	533
niietcm4_gpio.c	Файл содержит реализацию всех функции для работы с модулями GPIO	539
niietcm4_gpio.h	Файл содержит все прототипы функций для GPIO	551
niietcm4_rcc.c	Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера	569
niietcm4_rcc.h	Файл содержит все прототипы функций для RCC (Reset & Clock Control)	579
niietcm4_rtc.c	Файл содержит реализацию всех функции для работы с RTC	596

nietcm4_rtc.h	Файл содержит все прототипы функций для таймеров	597
nietcm4_timer.c	Файл содержит реализацию всех функции для работы с таймерами	601
nietcm4_timer.h	Файл содержит все прототипы функций для таймеров	605
nietcm4_uart.c	Файл содержит реализацию всех функции для работы с модулями UART	612
nietcm4_uart.h	Файл содержит все прототипы функций для UART	621
nietcm4_userflash.c	Файл содержит реализацию всех функции для работы с пользовательской флеш .	636
nietcm4_userflash.h	Файл содержит все прототипы функций для пользовательской флеш	640
nietcm4_watchdog.c	Файл содержит реализацию всех функции для работы со сторожевым таймером .	645
nietcm4_watchdog.h	Файл содержит все прототипы функций для сторожевого таймера	649

Глава 6

Группы

6.1 Константы

Группы

- Маски каналов для измерений
- Маски выбора цифровых компараторов
- Маски выбора секвенсоров

6.1.1 Подробное описание

6.2 Маски каналов для измерений

Макросы

```

• #define ADC_Channel_None ((uint32_t)0x00000000)
• #define ADC_Channel_0 ((uint32_t)0x00000001)
• #define ADC_Channel_1 ((uint32_t)0x00000002)
• #define ADC_Channel_2 ((uint32_t)0x00000004)
• #define ADC_Channel_3 ((uint32_t)0x00000008)
• #define ADC_Channel_4 ((uint32_t)0x00000010)
• #define ADC_Channel_5 ((uint32_t)0x00000020)
• #define ADC_Channel_6 ((uint32_t)0x00000040)
• #define ADC_Channel_7 ((uint32_t)0x00000080)
• #define ADC_Channel_8 ((uint32_t)0x00000100)
• #define ADC_Channel_9 ((uint32_t)0x00000200)
• #define ADC_Channel_10 ((uint32_t)0x00000400)
• #define ADC_Channel_11 ((uint32_t)0x00000800)
• #define ADC_Channel_12 ((uint32_t)0x00001000)
• #define ADC_Channel_13 ((uint32_t)0x00002000)
• #define ADC_Channel_14 ((uint32_t)0x00004000)
• #define ADC_Channel_15 ((uint32_t)0x00008000)
• #define ADC_Channel_16 ((uint32_t)0x00010000)
• #define ADC_Channel_17 ((uint32_t)0x00020000)
• #define ADC_Channel_18 ((uint32_t)0x00040000)
• #define ADC_Channel_19 ((uint32_t)0x00080000)
• #define ADC_Channel_20 ((uint32_t)0x00100000)
• #define ADC_Channel_21 ((uint32_t)0x00200000)
• #define ADC_Channel_22 ((uint32_t)0x00400000)
• #define ADC_Channel_23 ((uint32_t)0x00800000)
• #define ADC_Channel_All ((uint32_t)0x00FFFFFF)
• #define IS_ADC_CHANNEL(CHANNEL) (((CHANNEL) & (uint32_t)0xFF000000) ==
((uint32_t)0x00000000))

```

Макрос проверки попадания масок каналов в допустимый диапазон.

6.2.1 Подробное описание

6.2.2 Макросы

6.2.2.1 #define ADC_Channel_0 ((uint32_t)0x00000001)

Канал ADC 0

См. определение в файле niietcm4_adc.h строка 57

6.2.2.2 #define ADC_Channel_1 ((uint32_t)0x00000002)

Канал ADC 1

См. определение в файле niietcm4_adc.h строка 58

6.2.2.3 #define ADC_Channel_10 ((uint32_t)0x00000400)

Канал ADC 10

См. определение в файле niietcm4_adc.h строка 67

6.2.2.4 `#define ADC_Channel_11 ((uint32_t)0x00000800)`

Канал ADC 11

См. определение в файле `niietcm4_adc.h` строка 68

6.2.2.5 `#define ADC_Channel_12 ((uint32_t)0x00001000)`

Канал ADC 12

См. определение в файле `niietcm4_adc.h` строка 69

6.2.2.6 `#define ADC_Channel_13 ((uint32_t)0x00002000)`

Канал ADC 13

См. определение в файле `niietcm4_adc.h` строка 70

6.2.2.7 `#define ADC_Channel_14 ((uint32_t)0x00004000)`

Канал ADC 14

См. определение в файле `niietcm4_adc.h` строка 71

6.2.2.8 `#define ADC_Channel_15 ((uint32_t)0x00008000)`

Канал ADC 15

См. определение в файле `niietcm4_adc.h` строка 72

6.2.2.9 `#define ADC_Channel_16 ((uint32_t)0x00010000)`

Канал ADC 16

См. определение в файле `niietcm4_adc.h` строка 73

6.2.2.10 `#define ADC_Channel_17 ((uint32_t)0x00020000)`

Канал ADC 17

См. определение в файле `niietcm4_adc.h` строка 74

6.2.2.11 `#define ADC_Channel_18 ((uint32_t)0x00040000)`

Канал ADC 18

См. определение в файле `niietcm4_adc.h` строка 75

6.2.2.12 `#define ADC_Channel_19 ((uint32_t)0x00080000)`

Канал ADC 19

См. определение в файле `niietcm4_adc.h` строка 76

6.2.2.13 `#define ADC_Channel_2 ((uint32_t)0x00000004)`

Канал ADC 2

См. определение в файле niietcm4_adc.h строка 59

6.2.2.14 `#define ADC_Channel_20 ((uint32_t)0x00100000)`

Канал ADC 20

См. определение в файле niietcm4_adc.h строка 77

6.2.2.15 `#define ADC_Channel_21 ((uint32_t)0x00200000)`

Канал ADC 21

См. определение в файле niietcm4_adc.h строка 78

6.2.2.16 `#define ADC_Channel_22 ((uint32_t)0x00400000)`

Канал ADC 22

См. определение в файле niietcm4_adc.h строка 79

6.2.2.17 `#define ADC_Channel_23 ((uint32_t)0x00800000)`

Канал ADC 23

См. определение в файле niietcm4_adc.h строка 80

6.2.2.18 `#define ADC_Channel_3 ((uint32_t)0x00000008)`

Канал ADC 3

См. определение в файле niietcm4_adc.h строка 60

6.2.2.19 `#define ADC_Channel_4 ((uint32_t)0x00000010)`

Канал ADC 4

См. определение в файле niietcm4_adc.h строка 61

6.2.2.20 `#define ADC_Channel_5 ((uint32_t)0x00000020)`

Канал ADC 5

См. определение в файле niietcm4_adc.h строка 62

6.2.2.21 `#define ADC_Channel_6 ((uint32_t)0x00000040)`

Канал ADC 6

См. определение в файле niietcm4_adc.h строка 63

6.2.2.22 `#define ADC_Channel_7 ((uint32_t)0x00000080)`

Канал ADC 7

См. определение в файле niietcm4_adc.h строка 64

6.2.2.23 `#define ADC_Channel_8 ((uint32_t)0x00000100)`

Канал ADC 8

См. определение в файле `niietcm4_adc.h` строка 65

6.2.2.24 `#define ADC_Channel_9 ((uint32_t)0x00000200)`

Канал ADC 9

См. определение в файле `niietcm4_adc.h` строка 66

6.2.2.25 `#define ADC_Channel_All ((uint32_t)0x00FFFFFF)`

Все каналы

См. определение в файле `niietcm4_adc.h` строка 81

6.2.2.26 `#define ADC_Channel_None ((uint32_t)0x00000000)`

Канал ADC не выбран

См. определение в файле `niietcm4_adc.h` строка 56

Используется в `ADC_SEQ_StructInit()`.

6.3 Маски выбора цифровых компараторов

Макросы

- `#define ADC_DC_None ((uint32_t)0x00000000)`
- `#define ADC_DC_0 ((uint32_t)0x00000001)`
- `#define ADC_DC_1 ((uint32_t)0x00000002)`
- `#define ADC_DC_2 ((uint32_t)0x00000004)`
- `#define ADC_DC_3 ((uint32_t)0x00000008)`
- `#define ADC_DC_4 ((uint32_t)0x00000010)`
- `#define ADC_DC_5 ((uint32_t)0x00000020)`
- `#define ADC_DC_6 ((uint32_t)0x00000040)`
- `#define ADC_DC_7 ((uint32_t)0x00000080)`
- `#define ADC_DC_8 ((uint32_t)0x00000100)`
- `#define ADC_DC_9 ((uint32_t)0x00000200)`
- `#define ADC_DC_10 ((uint32_t)0x00000400)`
- `#define ADC_DC_11 ((uint32_t)0x00000800)`
- `#define ADC_DC_12 ((uint32_t)0x00001000)`
- `#define ADC_DC_13 ((uint32_t)0x00002000)`
- `#define ADC_DC_14 ((uint32_t)0x00004000)`
- `#define ADC_DC_15 ((uint32_t)0x00008000)`
- `#define ADC_DC_16 ((uint32_t)0x00010000)`
- `#define ADC_DC_17 ((uint32_t)0x00020000)`
- `#define ADC_DC_18 ((uint32_t)0x00040000)`
- `#define ADC_DC_19 ((uint32_t)0x00080000)`
- `#define ADC_DC_20 ((uint32_t)0x00100000)`
- `#define ADC_DC_21 ((uint32_t)0x00200000)`
- `#define ADC_DC_22 ((uint32_t)0x00400000)`
- `#define ADC_DC_23 ((uint32_t)0x00800000)`
- `#define ADC_DC_All ((uint32_t)0x00FFFFFF)`
- `#define IS_ADC_DC(DC) (((DC) & (uint32_t)0xFF000000) == ((uint32_t)0x00000000))`

Макрос проверки попадания масок компараторов в допустимый диапазон.

6.3.1 Подробное описание

6.3.2 Макросы

6.3.2.1 `#define ADC_DC_0 ((uint32_t)0x00000001)`

Цифровой компаратор 0

См. определение в файле `niietcm4_adc.h` строка 97

6.3.2.2 `#define ADC_DC_1 ((uint32_t)0x00000002)`

Цифровой компаратор 1

См. определение в файле `niietcm4_adc.h` строка 98

6.3.2.3 `#define ADC_DC_10 ((uint32_t)0x00000400)`

Цифровой компаратор 10

См. определение в файле `niietcm4_adc.h` строка 107

6.3.2.4 `#define ADC_DC_11 ((uint32_t)0x00000800)`

Цифровой компаратор 11

См. определение в файле `niietcm4_adc.h` строка 108

6.3.2.5 `#define ADC_DC_12 ((uint32_t)0x00001000)`

Цифровой компаратор 12

См. определение в файле `niietcm4_adc.h` строка 109

6.3.2.6 `#define ADC_DC_13 ((uint32_t)0x00002000)`

Цифровой компаратор 13

См. определение в файле `niietcm4_adc.h` строка 110

6.3.2.7 `#define ADC_DC_14 ((uint32_t)0x00004000)`

Цифровой компаратор 14

См. определение в файле `niietcm4_adc.h` строка 111

6.3.2.8 `#define ADC_DC_15 ((uint32_t)0x00008000)`

Цифровой компаратор 15

См. определение в файле `niietcm4_adc.h` строка 112

6.3.2.9 `#define ADC_DC_16 ((uint32_t)0x00010000)`

Цифровой компаратор 16

См. определение в файле `niietcm4_adc.h` строка 113

6.3.2.10 `#define ADC_DC_17 ((uint32_t)0x00020000)`

Цифровой компаратор 17

См. определение в файле `niietcm4_adc.h` строка 114

6.3.2.11 `#define ADC_DC_18 ((uint32_t)0x00040000)`

Цифровой компаратор 18

См. определение в файле `niietcm4_adc.h` строка 115

6.3.2.12 `#define ADC_DC_19 ((uint32_t)0x00080000)`

Цифровой компаратор 19

См. определение в файле `niietcm4_adc.h` строка 116

6.3.2.13 `#define ADC_DC_2 ((uint32_t)0x00000004)`

Цифровой компаратор 2

См. определение в файле niietcm4_adc.h строка 99

6.3.2.14 `#define ADC_DC_20 ((uint32_t)0x00100000)`

Цифровой компаратор 20

См. определение в файле niietcm4_adc.h строка 117

6.3.2.15 `#define ADC_DC_21 ((uint32_t)0x00200000)`

Цифровой компаратор 21

См. определение в файле niietcm4_adc.h строка 118

6.3.2.16 `#define ADC_DC_22 ((uint32_t)0x00400000)`

Цифровой компаратор 22

См. определение в файле niietcm4_adc.h строка 119

6.3.2.17 `#define ADC_DC_23 ((uint32_t)0x00800000)`

Цифровой компаратор 23

См. определение в файле niietcm4_adc.h строка 120

6.3.2.18 `#define ADC_DC_3 ((uint32_t)0x00000008)`

Цифровой компаратор 3

См. определение в файле niietcm4_adc.h строка 100

6.3.2.19 `#define ADC_DC_4 ((uint32_t)0x00000010)`

Цифровой компаратор 4

См. определение в файле niietcm4_adc.h строка 101

6.3.2.20 `#define ADC_DC_5 ((uint32_t)0x00000020)`

Цифровой компаратор 5

См. определение в файле niietcm4_adc.h строка 102

6.3.2.21 `#define ADC_DC_6 ((uint32_t)0x00000040)`

Цифровой компаратор 6

См. определение в файле niietcm4_adc.h строка 103

6.3.2.22 `#define ADC_DC_7 ((uint32_t)0x00000080)`

Цифровой компаратор 7

См. определение в файле niietcm4_adc.h строка 104

6.3.2.23 `#define ADC_DC_8 ((uint32_t)0x00000100)`

Цифровой компаратор 8

См. определение в файле `niietcm4_adc.h` строка 105

6.3.2.24 `#define ADC_DC_9 ((uint32_t)0x00000200)`

Цифровой компаратор 9

См. определение в файле `niietcm4_adc.h` строка 106

6.3.2.25 `#define ADC_DC_All ((uint32_t)0x00FFFFFF)`

Все цифровые компараторы

См. определение в файле `niietcm4_adc.h` строка 121

6.3.2.26 `#define ADC_DC_None ((uint32_t)0x00000000)`

Цифровой компаратор не выбран

См. определение в файле `niietcm4_adc.h` строка 96

Используется в `ADC_SEQ_StructInit()`.

6.4.2.7 `#define ADC_SEQ_6 ((uint32_t)0x00000040)`

Секвенсор 6

См. определение в файле `niietcm4_adc.h` строка 143

6.4.2.8 `#define ADC_SEQ_7 ((uint32_t)0x00000080)`

Секвенсор 7

См. определение в файле `niietcm4_adc.h` строка 144

6.5 Типы

Структуры данных

- struct `ADC_Init_TypeDef`
Структура инициализации модулей АЦП
- struct `ADC_DC_Init_TypeDef`
Структура инициализации цифровых компараторов.
- struct `ADC_SEQ_Init_TypeDef`
Структура инициализации секвенсоров.

Макросы

- `#define IS_ADC_SEQ_IT_RATE(IT_RATE) (((IT_RATE) > ((uint32_t)0x0)) && ((IT_RATE) < ((uint32_t)0x100)))`
Проверка значения количества перезапусков модулей АЦП секвенсором после которого генерируется прерывание, на попадание в допустимый диапазон.
- `#define IS_ADC_SEQ_CONVERSION_COUNT(CONVERSION_COUNT) (((CONVERSION_COUNT) > ((uint32_t)0x0)) && ((CONVERSION_COUNT) <= ((uint32_t)0x100)))`
Проверка значения количества перезапусков модулей АЦП секвенсором после запуска секвенсора по событию на попадание в допустимый диапазон.
- `#define IS_ADC_SEQ_CONVERSION_DELAY(CONVERSION_DELAY) ((CONVERSION_DELAY) < ((uint32_t)0x1000000))`
Проверка значения задержки запуска преобразования модулем АЦП на попадание в допустимый диапазон.
- `#define IS_ADC_PHASE(PHASE) ((PHASE) < ((uint32_t)0x1000))`
Проверка значения задержки начала преобразования модулем АЦП после запуска модуля секвенсором на попадание в допустимый диапазон.
- `#define IS_ADC_DC_THRESHOLD(THRESHOLD) ((THRESHOLD) < ((uint32_t)0x1000))`
Проверка значения порога диапазона срабатывания компаратора на попадание в допустимый диапазон.
- `#define IS_ADC_SEQ_START_EVENT(START_EVENT)`
Макрос проверки аргументов типа `ADC_SEQ_StartEvent_TypeDef`.
- `#define IS_ADC_AVERAGE(AVERAGE)`
Макрос проверки аргументов типа `ADC_Average_TypeDef`.
- `#define IS_ADC_DC_CHANNEL(CHANNEL)`
Макрос проверки аргументов типа `ADC_DC_Channel_TypeDef`.
- `#define IS_ADC_DC_MODE(MODE)`
Макрос проверки аргументов типа `ADC_DC_Mode_TypeDef`.
- `#define IS_ADC_DC_CONDITION(CONDITION)`
Макрос проверки аргументов типа `ADC_DC_Condition_TypeDef`.
- `#define IS_ADC_SEQ_FIFO_LEVEL(FIFO_LEVEL)`
Макрос проверки аргументов типа `ADC_SEQ_FIFOLevel_TypeDef`.
- `#define IS_ADC_DC_MODULE(MODULE)`
Макрос проверки аргументов типа `ADC_DC_Module_TypeDef`.
- `#define IS_ADC_SEQ_MODULE(MODULE)`
Макрос проверки аргументов типа `ADC_SEQ_Module_TypeDef`.
- `#define IS_ADC_MODULE(MODULE)`
Макрос проверки аргументов типа `ADC_Module_TypeDef`.
- `#define IS_ADC_RESOLUTION(RESOLUTION)`
Макрос проверки аргументов типа `ADC_Resolution_TypeDef`.
- `#define IS_ADC_MEASURE(MEASURE)`

- Макрос проверки аргументов типа `ADC_Measure_TypeDef`.
- `#define IS_ADC_MODE(MODE)`
Макрос проверки аргументов типа `ADC_Mode_TypeDef`.

Перечисления

- `enum ADC_SEQ_StartEvent_TypeDef {`
`ADC_SEQ_StartEvent_SWReq = ((uint32_t)0x0), ADC_SEQ_StartEvent_CMP0 =`
`((uint32_t)0x1), ADC_SEQ_StartEvent_CMP1 = ((uint32_t)0x2), ADC_SEQ_StartEvent_↵`
`CMP2 = ((uint32_t)0x3),`
`ADC_SEQ_StartEvent_ITGPIO = ((uint32_t)0x4), ADC_SEQ_StartEvent_TIM = ((uint32_↵`
`t)0x5), ADC_SEQ_StartEvent_PWM0 = ((uint32_t)0x6), ADC_SEQ_StartEvent_PWM1 =`
`((uint32_t)0x7),`
`ADC_SEQ_StartEvent_PWM2 = ((uint32_t)0x8), ADC_SEQ_StartEvent_PWM3 =`
`((uint32_t)0x9), ADC_SEQ_StartEvent_PWM4 = ((uint32_t)0xA), ADC_SEQ_StartEvent_↵`
`PWM5 = ((uint32_t)0xB),`
`ADC_SEQ_StartEvent_Cycle = ((uint32_t)0xF) }`
События запуска секвенсоров.
- `enum ADC_Average_TypeDef {`
`ADC_Average_Disable, ADC_Average_2, ADC_Average_4, ADC_Average_8,`
`ADC_Average_16, ADC_Average_32, ADC_Average_64 }`
Количество измерений, используемых для получения результата преобразования.
- `enum ADC_DC_Channel_TypeDef {`
`ADC_DC_Channel_0, ADC_DC_Channel_1, ADC_DC_Channel_2, ADC_DC_Channel_3,`
`ADC_DC_Channel_4, ADC_DC_Channel_5, ADC_DC_Channel_6, ADC_DC_Channel_7,`
`ADC_DC_Channel_8, ADC_DC_Channel_9, ADC_DC_Channel_10, ADC_DC_Channel_↵`
`11,`
`ADC_DC_Channel_12, ADC_DC_Channel_13, ADC_DC_Channel_14, ADC_DC_↵`
`Channel_15,`
`ADC_DC_Channel_16, ADC_DC_Channel_17, ADC_DC_Channel_18, ADC_DC_↵`
`Channel_19,`
`ADC_DC_Channel_20, ADC_DC_Channel_21, ADC_DC_Channel_22, ADC_DC_↵`
`Channel_23,`
`ADC_DC_Channel_None }`
Выбор канала, подключаемого к цифровому компаратору.
- `enum ADC_DC_Mode_TypeDef { ADC_DC_Mode_Multiple, ADC_DC_Mode_Single, AD_↵`
`C_DC_Mode_MultipleHyst, ADC_DC_Mode_SingleHyst }`
Режим срабатывания компаратора.
- `enum ADC_DC_Condition_TypeDef { ADC_DC_Condition_Low = ((uint32_t)0), ADC_D_↵`
`C_Condition_Window = ((uint32_t)1), ADC_DC_Condition_High = ((uint32_t)3) }`
Условие срабатывания компаратора.
- `enum ADC_SEQ_FIFOLevel_TypeDef {`
`ADC_SEQ_FIFOLevel_1 = ((uint32_t)1), ADC_SEQ_FIFOLevel_2 = ((uint32_t)2), ADC_↵`
`_SEQ_FIFOLevel_4 = ((uint32_t)3), ADC_SEQ_FIFOLevel_8 = ((uint32_t)4),`
`ADC_SEQ_FIFOLevel_16 = ((uint32_t)5), ADC_SEQ_FIFOLevel_32 = ((uint32_t)6) }`
Количество результатов измерений записанных в буфер секвенсора, по достижению которого вы-
зывается DMA.
- `enum ADC_DC_Module_TypeDef {`
`ADC_DC_Module_0, ADC_DC_Module_1, ADC_DC_Module_2, ADC_DC_Module_3,`
`ADC_DC_Module_4, ADC_DC_Module_5, ADC_DC_Module_6, ADC_DC_Module_7,`
`ADC_DC_Module_8, ADC_DC_Module_9, ADC_DC_Module_10, ADC_DC_Module_11,`
`ADC_DC_Module_12, ADC_DC_Module_13, ADC_DC_Module_14, ADC_DC_Module_↵`
`15,`
`ADC_DC_Module_16, ADC_DC_Module_17, ADC_DC_Module_18, ADC_DC_Module_↵`
`19,`
`ADC_DC_Module_20, ADC_DC_Module_21, ADC_DC_Module_22, ADC_DC_Module_23`
`}`

- Выбор модуля цифрового компаратора.
- enum `ADC_SEQ_Module_TypeDef` {
`ADC_SEQ_Module_0`, `ADC_SEQ_Module_1`, `ADC_SEQ_Module_2`, `ADC_SEQ_Module_3`,
`ADC_SEQ_Module_4`, `ADC_SEQ_Module_5`, `ADC_SEQ_Module_6`, `ADC_SEQ_Module_7` }
- Выбор модуля секвенсора.
- enum `ADC_Module_TypeDef` {
`ADC_Module_0`, `ADC_Module_1`, `ADC_Module_2`, `ADC_Module_3`,
`ADC_Module_4`, `ADC_Module_5`, `ADC_Module_6`, `ADC_Module_7`,
`ADC_Module_8`, `ADC_Module_9`, `ADC_Module_10`, `ADC_Module_11` }
- Выбор модуля АЦП.
- enum `ADC_Resolution_TypeDef` { `ADC_Resolution_12bit`, `ADC_Resolution_10bit` }
- Выбор разрядности модуля АЦП.
- enum `ADC_Measure_TypeDef` { `ADC_Measure_Single`, `ADC_Measure_Diff` }
- Выбор режима работы АЦП.
- enum `ADC_Mode_TypeDef` { `ADC_Mode_Powerdown` = ((uint32_t)0), `ADC_Mode_StandBy` = ((uint32_t)1), `ADC_Mode_Active` = ((uint32_t)3) }
- Выбор режима работы АЦП.

6.5.1 Подробное описание

6.5.2 Макросы

6.5.2.1 #define IS_ADC_AVERAGE(AVERAGE)

Макроопределение:

```
((AVERAGE) == ADC_Average_Disable) || \
((AVERAGE) == ADC_Average_2)         || \
((AVERAGE) == ADC_Average_4)         || \
((AVERAGE) == ADC_Average_8)         || \
((AVERAGE) == ADC_Average_16)        || \
((AVERAGE) == ADC_Average_32)        || \
((AVERAGE) == ADC_Average_64))
```

Макрос проверки аргументов типа `ADC_Average_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 255

Используется в `ADC_Init()`.

6.5.2.2 #define IS_ADC_DC_CHANNEL(CHANNEL)

Макроопределение:

```
((CHANNEL) == ADC_DC_Channel_0) || \
((CHANNEL) == ADC_DC_Channel_1) || \
((CHANNEL) == ADC_DC_Channel_2) || \
((CHANNEL) == ADC_DC_Channel_3) || \
((CHANNEL) == ADC_DC_Channel_4) || \
((CHANNEL) == ADC_DC_Channel_5) || \
((CHANNEL) == ADC_DC_Channel_6) || \
((CHANNEL) == ADC_DC_Channel_7) || \
((CHANNEL) == ADC_DC_Channel_8) || \
((CHANNEL) == ADC_DC_Channel_9) || \
((CHANNEL) == ADC_DC_Channel_10) || \
((CHANNEL) == ADC_DC_Channel_11) || \
((CHANNEL) == ADC_DC_Channel_12) || \
((CHANNEL) == ADC_DC_Channel_13) || \
((CHANNEL) == ADC_DC_Channel_14) || \
((CHANNEL) == ADC_DC_Channel_15) || \
((CHANNEL) == ADC_DC_Channel_16) || \
((CHANNEL) == ADC_DC_Channel_17) || \
```

```
((CHANNEL) == ADC_DC_Channel_18) || \
((CHANNEL) == ADC_DC_Channel_19) || \
((CHANNEL) == ADC_DC_Channel_20) || \
((CHANNEL) == ADC_DC_Channel_21) || \
((CHANNEL) == ADC_DC_Channel_22) || \
((CHANNEL) == ADC_DC_Channel_23) || \
((CHANNEL) == ADC_DC_Channel_None))
```

Макрос проверки аргументов типа [ADC_DC_Channel_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 300

Используется в ADC_DC_Init().

6.5.2.3 #define IS_ADC_DC_CONDITION(CONDITION)

Макроопределение:

```
((CONDITION) == ADC_DC_Condition_Low) || \
((CONDITION) == ADC_DC_Condition_Window) || \
((CONDITION) == ADC_DC_Condition_High))
```

Макрос проверки аргументов типа [ADC_DC_Condition_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 362

Используется в ADC_DC_Init() и ADC_DC_ITConfig().

6.5.2.4 #define IS_ADC_DC_MODE(MODE)

Макроопределение:

```
((MODE) == ADC_DC_Mode_Single) || \
((MODE) == ADC_DC_Mode_Multiple) || \
((MODE) == ADC_DC_Mode_SingleHyst) || \
((MODE) == ADC_DC_Mode_MultipleHyst))
```

Макрос проверки аргументов типа [ADC_DC_Mode_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 342

Используется в ADC_DC_Init() и ADC_DC_ITConfig().

6.5.2.5 #define IS_ADC_DC_MODULE(MODULE)

Макроопределение:

```
((MODULE) == ADC_DC_Module_0) || \
((MODULE) == ADC_DC_Module_1) || \
((MODULE) == ADC_DC_Module_2) || \
((MODULE) == ADC_DC_Module_3) || \
((MODULE) == ADC_DC_Module_4) || \
((MODULE) == ADC_DC_Module_5) || \
((MODULE) == ADC_DC_Module_6) || \
((MODULE) == ADC_DC_Module_7) || \
((MODULE) == ADC_DC_Module_8) || \
((MODULE) == ADC_DC_Module_9) || \
((MODULE) == ADC_DC_Module_10) || \
((MODULE) == ADC_DC_Module_11) || \
((MODULE) == ADC_DC_Module_12) || \
((MODULE) == ADC_DC_Module_13) || \
((MODULE) == ADC_DC_Module_14) || \
((MODULE) == ADC_DC_Module_15) || \
((MODULE) == ADC_DC_Module_16) || \
((MODULE) == ADC_DC_Module_17) || \
((MODULE) == ADC_DC_Module_18) || \
((MODULE) == ADC_DC_Module_19) || \
((MODULE) == ADC_DC_Module_20) || \
((MODULE) == ADC_DC_Module_21) || \
((MODULE) == ADC_DC_Module_22) || \
((MODULE) == ADC_DC_Module_23))
```

Макрос проверки аргументов типа [ADC_DC_Module_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 427

Используется в `ADC_DC_Cmd()`, `ADC_DC_DeInit()`, `ADC_DC_GetLastData()`, `ADC_DC_ITCmd()`, `ADC_DC_ITConfig()`, `ADC_DC_ITGenCmd()`, `ADC_DC_ITMaskCmd()`, `ADC_DC_ITMaskedStatus()`, `ADC_DC_ITRawStatus()`, `ADC_DC_ITStatusClear()`, `ADC_DC_TrigStatus()` и `ADC_DC_TrigStatusClear()`.

6.5.2.6 `#define IS_ADC_MEASURE(MEASURE)`

Макроопределение:

```
((MEASURE) == ADC_Measure_Single) || \
((MEASURE) == ADC_Measure_Diff)
```

Макрос проверки аргументов типа [ADC_Measure_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 549

Используется в `ADC_Init()`.

6.5.2.7 `#define IS_ADC_MODE(MODE)`

Макроопределение:

```
((MODE) == ADC_Mode_Powerdown) || \
((MODE) == ADC_Mode_StandBy) || \
((MODE) == ADC_Mode_Active)
```

Макрос проверки аргументов типа [ADC_Mode_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 567

Используется в `ADC_Init()`.

6.5.2.8 `#define IS_ADC_MODULE(MODULE)`

Макроопределение:

```
((MODULE) == ADC_Module_0) || \
((MODULE) == ADC_Module_1) || \
((MODULE) == ADC_Module_2) || \
((MODULE) == ADC_Module_3) || \
((MODULE) == ADC_Module_4) || \
((MODULE) == ADC_Module_5) || \
((MODULE) == ADC_Module_6) || \
((MODULE) == ADC_Module_7) || \
((MODULE) == ADC_Module_8) || \
((MODULE) == ADC_Module_9) || \
((MODULE) == ADC_Module_10) || \
((MODULE) == ADC_Module_11)
```

Макрос проверки аргументов типа [ADC_Module_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 505

Используется в `ADC_Cmd()`, `ADC_DeInit()` и `ADC_Init()`.

6.5.2.9 `#define IS_ADC_RESOLUTION(RESOLUTION)`

Макроопределение:

```
((RESOLUTION) == ADC_Resolution_12bit) || \
((RESOLUTION) == ADC_Resolution_10bit)
```

Макрос проверки аргументов типа `ADC_Resolution_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 532

Используется в `ADC_Init()`.

6.5.2.10 `#define IS_ADC_SEQ_FIFO_LEVEL(FIFO_LEVEL)`

Макроопределение:

```
((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_1) || \
    ((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_2) || \
    ((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_4) || \
    ((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_8) || \
    ((FIFO_LEVEL) == \
    ADC_SEQ_FIFOLevel_16) || \
    ((FIFO_LEVEL) == \
    ADC_SEQ_FIFOLevel_32))
```

Макрос проверки аргументов типа `ADC_SEQ_FIFOLevel_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 384

Используется в `ADC_SEQ_DMAConfig()`.

6.5.2.11 `#define IS_ADC_SEQ_MODULE(MODULE)`

Макроопределение:

```
((MODULE) == ADC_SEQ_Module_0) || \
    ((MODULE) == ADC_SEQ_Module_1) || \
    ((MODULE) == ADC_SEQ_Module_2) || \
    ((MODULE) == ADC_SEQ_Module_3) || \
    ((MODULE) == ADC_SEQ_Module_4) || \
    ((MODULE) == ADC_SEQ_Module_5) || \
    ((MODULE) == ADC_SEQ_Module_6) || \
    ((MODULE) == ADC_SEQ_Module_7))
```

Макрос проверки аргументов типа `ADC_SEQ_Module_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 472

Используется в `ADC_SEQ_Cmd()`, `ADC_SEQ_DeInit()`, `ADC_SEQ_DMAMCmd()`, `ADC_SEQ_DMAConfig()`, `ADC_SEQ_DMAErrorStatus()`, `ADC_SEQ_DMAErrorStatusClear()`, `ADC_SEQ_FIFOEmptyStatus()`, `ADC_SEQ_FIFOEmptyStatusClear()`, `ADC_SEQ_FIFOFullStatus()`, `ADC_SEQ_FIFOFullStatusClear()`, `ADC_SEQ_GetConversionCount()`, `ADC_SEQ_GetFIFOData()`, `ADC_SEQ_GetFIFOLoad()`, `ADC_SEQ_GetITCount()`, `ADC_SEQ_Init()`, `ADC_SEQ_ITCmd()`, `ADC_SEQ_ITConfig()`, `ADC_SEQ_ITCountRst()`, `ADC_SEQ_ITMaskedStatus()`, `ADC_SEQ_ITRawStatus()` и `ADC_SEQ_ITStatusClear()`.

6.5.2.12 `#define IS_ADC_SEQ_START_EVENT(START_EVENT)`

Макроопределение:

```
((START_EVENT) == ADC_SEQ_StartEvent_SWReq) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_CMP0) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_CMP1) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_CMP2) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_ITGPIO) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_TIM) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_PWM0) || \
    ((START_EVENT) == \
    ADC_SEQ_StartEvent_PWM1) || \
```

```

((START_EVENT) ==
ADC_SEQ_StartEvent_PWM2) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM3) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM4) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM5) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_Cycle))

```

Макрос проверки аргументов типа `ADC_SEQ_StartEvent_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 222

Используется в `ADC_SEQ_Init()`.

6.5.3 Перечисления

6.5.3.1 enum ADC_Average_TypeDef

Количество измерений, используемых для получения результата преобразования.

Элементы перечислений

```

ADC_Average_Disable  Усреднители не используются.
ADC_Average_2        Усреднение по 2 измерениям.
ADC_Average_4        Усреднение по 4 измерениям.
ADC_Average_8        Усреднение по 8 измерениям.
ADC_Average_16       Усреднение по 16 измерениям.
ADC_Average_32       Усреднение по 32 измерениям.
ADC_Average_64       Усреднение по 64 измерениям.

```

См. определение в файле `niietcm4_adc.h` строка 240

6.5.3.2 enum ADC_DC_Channel_TypeDef

Выбор канала, подключаемого к цифровому компаратору.

Элементы перечислений

```

ADC_DC_Channel_0  Результат с канала 0 будет передан на компаратор.
ADC_DC_Channel_1  Результат с канала 1 будет передан на компаратор.
ADC_DC_Channel_2  Результат с канала 2 будет передан на компаратор.
ADC_DC_Channel_3  Результат с канала 3 будет передан на компаратор.
ADC_DC_Channel_4  Результат с канала 4 будет передан на компаратор.
ADC_DC_Channel_5  Результат с канала 5 будет передан на компаратор.
ADC_DC_Channel_6  Результат с канала 6 будет передан на компаратор.
ADC_DC_Channel_7  Результат с канала 7 будет передан на компаратор.
ADC_DC_Channel_8  Результат с канала 8 будет передан на компаратор.
ADC_DC_Channel_9  Результат с канала 9 будет передан на компаратор.
ADC_DC_Channel_10 Результат с канала 10 будет передан на компаратор.
ADC_DC_Channel_11 Результат с канала 11 будет передан на компаратор.
ADC_DC_Channel_12 Результат с канала 12 будет передан на компаратор.
ADC_DC_Channel_13 Результат с канала 13 будет передан на компаратор.

```


ADC_DC_Channel_14 Результат с канала 14 будет передан на компаратор.
 ADC_DC_Channel_15 Результат с канала 15 будет передан на компаратор.
 ADC_DC_Channel_16 Результат с канала 16 будет передан на компаратор.
 ADC_DC_Channel_17 Результат с канала 17 будет передан на компаратор.
 ADC_DC_Channel_18 Результат с канала 18 будет передан на компаратор.
 ADC_DC_Channel_19 Результат с канала 19 будет передан на компаратор.
 ADC_DC_Channel_20 Результат с канала 20 будет передан на компаратор.
 ADC_DC_Channel_21 Результат с канала 21 будет передан на компаратор.
 ADC_DC_Channel_22 Результат с канала 22 будет передан на компаратор.
 ADC_DC_Channel_23 Результат с канала 23 будет передан на компаратор.
 ADC_DC_Channel_None Ни один из каналов не подключен к компаратору.

См. определение в файле niietcm4_adc.h строка 267

6.5.3.3 enum ADC_DC_Condition_TypeDef

Условие срабатывания компаратора.

Элементы перечислений

ADC_DC_Condition_Low Результат меньше либо равен нижней границе.
 ADC_DC_Condition_Window Результат внутри диапазона, задаваемого границами, либо равен одной из них.
 ADC_DC_Condition_High Результат выше либо равен верхней границе.

См. определение в файле niietcm4_adc.h строка 351

6.5.3.4 enum ADC_DC_Mode_TypeDef

Режим срабатывания компаратора.

Элементы перечислений

ADC_DC_Mode_Multiple Многократный.
 ADC_DC_Mode_Single Однократный.
 ADC_DC_Mode_MultipleHyst Многократный с гистерезисом.
 ADC_DC_Mode_SingleHyst Однократный с гистерезисом.

См. определение в файле niietcm4_adc.h строка 330

6.5.3.5 enum ADC_DC_Module_TypeDef

Выбор модуля цифрового компаратора.

Элементы перечислений

ADC_DC_Module_0 Модуль цифрового компаратора 0.
 ADC_DC_Module_1 Модуль цифрового компаратора 1
 ADC_DC_Module_2 Модуль цифрового компаратора 2
 ADC_DC_Module_3 Модуль цифрового компаратора 3
 ADC_DC_Module_4 Модуль цифрового компаратора 4

ADC_DC_Module_5	Модуль цифрового компаратора 5
ADC_DC_Module_6	Модуль цифрового компаратора 6
ADC_DC_Module_7	Модуль цифрового компаратора 7
ADC_DC_Module_8	Модуль цифрового компаратора 8
ADC_DC_Module_9	Модуль цифрового компаратора 9
ADC_DC_Module_10	Модуль цифрового компаратора 10
ADC_DC_Module_11	Модуль цифрового компаратора 11
ADC_DC_Module_12	Модуль цифрового компаратора 12
ADC_DC_Module_13	Модуль цифрового компаратора 13
ADC_DC_Module_14	Модуль цифрового компаратора 14
ADC_DC_Module_15	Модуль цифрового компаратора 15
ADC_DC_Module_16	Модуль цифрового компаратора 16
ADC_DC_Module_17	Модуль цифрового компаратора 17
ADC_DC_Module_18	Модуль цифрового компаратора 18
ADC_DC_Module_19	Модуль цифрового компаратора 19
ADC_DC_Module_20	Модуль цифрового компаратора 20
ADC_DC_Module_21	Модуль цифрового компаратора 21
ADC_DC_Module_22	Модуль цифрового компаратора 22
ADC_DC_Module_23	Модуль цифрового компаратора 23

См. определение в файле niietcm4_adc.h строка 395

6.5.3.6 enum ADC_Measure_TypeDef

Выбор режима работы АЦП.

Элементы перечислений

ADC_Measure_Single	Однополярный режим измерения по каналу.
ADC_Measure_Diff	Дифференциальный режим с противоположным каналом.

См. определение в файле niietcm4_adc.h строка 539

6.5.3.7 enum ADC_Mode_TypeDef

Выбор режима работы АЦП.

Элементы перечислений

ADC_Mode_Powerdown	Модуль выключен.
ADC_Mode_StandBy	Режим ожидания.
ADC_Mode_Active	Модуль включен.

См. определение в файле niietcm4_adc.h строка 556

6.5.3.8 enum ADC_Module_TypeDef

Выбор модуля АЦП.

Элементы перечислений

ADC_Module_0 Модуль АЦП 0.
ADC_Module_1 Модуль АЦП 1
ADC_Module_2 Модуль АЦП 2
ADC_Module_3 Модуль АЦП 3
ADC_Module_4 Модуль АЦП 4
ADC_Module_5 Модуль АЦП 5
ADC_Module_6 Модуль АЦП 6
ADC_Module_7 Модуль АЦП 7
ADC_Module_8 Модуль АЦП 8
ADC_Module_9 Модуль АЦП 9
ADC_Module_10 Модуль АЦП 10
ADC_Module_11 Модуль АЦП 11

См. определение в файле niietcm4_adc.h строка 485

6.5.3.9 enum ADC_Resolution_TypeDef

Выбор разрядности модуля АЦП.

Элементы перечислений

ADC_Resolution_12bit Разрядность модуля 12 бит.
ADC_Resolution_10bit Разрядность модуля 10 бит

См. определение в файле niietcm4_adc.h строка 522

6.5.3.10 enum ADC_SEQ_FIFOLevel_TypeDef

Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA.

Элементы перечислений

ADC_SEQ_FIFOLevel_1 Запрос DMA после заполнения 1 ячейки в буфере.
ADC_SEQ_FIFOLevel_2 Запрос DMA после заполнения 2 ячеек в буфере.
ADC_SEQ_FIFOLevel_4 Запрос DMA после заполнения 4 ячеек в буфере.
ADC_SEQ_FIFOLevel_8 Запрос DMA после заполнения 8 ячеек в буфере.
ADC_SEQ_FIFOLevel_16 Запрос DMA после заполнения 16 ячеек в буфере.
ADC_SEQ_FIFOLevel_32 Запрос DMA после заполнения 32 ячеек в буфере.

См. определение в файле niietcm4_adc.h строка 370

6.5.3.11 enum ADC_SEQ_Module_TypeDef

Выбор модуля секвенсора.

Элементы перечислений

ADC_SEQ_Module_0 Севенсор 0.
ADC_SEQ_Module_1 Севенсор 1
ADC_SEQ_Module_2 Севенсор 2
ADC_SEQ_Module_3 Севенсор 3
ADC_SEQ_Module_4 Севенсор 4
ADC_SEQ_Module_5 Севенсор 5
ADC_SEQ_Module_6 Севенсор 6
ADC_SEQ_Module_7 Севенсор 7

См. определение в файле niietcm4_adc.h строка 456

6.5.3.12 enum ADC_SEQ_StartEvent_TypeDef

События запуска секвенсоров.

Элементы перечислений

ADC_SEQ_StartEvent_SWReq Запуск по программному запросу.
ADC_SEQ_StartEvent_CMP0 Сигнал от блока аналогового компаратора 0.
ADC_SEQ_StartEvent_CMP1 Сигнал от блока аналогового компаратора 1.
ADC_SEQ_StartEvent_CMP2 Сигнал от блока аналогового компаратора 2.
ADC_SEQ_StartEvent_ITGPIO Любое прерывание GPIO.
ADC_SEQ_StartEvent_TIM Сигнал от блока таймеров.
ADC_SEQ_StartEvent_PWM0 Сигнал от блока 0 ШИМ.
ADC_SEQ_StartEvent_PWM1 Сигнал от блока 1 ШИМ.
ADC_SEQ_StartEvent_PWM2 Сигнал от блока 2 ШИМ.
ADC_SEQ_StartEvent_PWM3 Сигнал от блока 3 ШИМ.
ADC_SEQ_StartEvent_PWM4 Сигнал от блока 4 ШИМ.
ADC_SEQ_StartEvent_PWM5 Сигнал от блока 5 ШИМ.
ADC_SEQ_StartEvent_Cycle Циклическая работа сразу после запуска секвенсора

См. определение в файле niietcm4_adc.h строка 201

6.6 Функции

Группы

- [Инициализация](#)
- [Конфигурация секвенсоров для DMA](#)
- [Конфигурация прерываний](#)

Функции

- void [ADC_Cmd](#) (ADC_Module_TypeDef ADC_Module, [FunctionalState](#) State)
Включение модуля АЦП.
- void [ADC_SEQ_Cmd](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, [FunctionalState](#) State)
Включение секвенсора.
- void [ADC_SEQ_SWReq](#) ()
Программный запуск измерений всех разрешенных секвенсоров.
- uint32_t [ADC_SEQ_GetFIFOData](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Получение результата измерений из буфера секвенсора.
- uint32_t [ADC_SEQ_GetConversionCount](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.
- uint32_t [ADC_SEQ_GetFIFOLoad](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Получение количества измерений, сохраненных в буфере секвенсора.
- [FlagStatus](#) [ADC_SEQ_FIFOFullStatus](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.
- void [ADC_SEQ_FIFOFullStatusClear](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Сброс флага заполнения буфера секвенсора.
- [FlagStatus](#) [ADC_SEQ_FIFOEmptyStatus](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.
- void [ADC_SEQ_FIFOEmptyStatusClear](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Сброс флага пустоты буфера секвенсора.
- void [ADC_DC_Cmd](#) (ADC_DC_Module_TypeDef ADC_DC_Module, [FunctionalState](#) State)
Включение выходного триггера цифрового компаратора.
- [FlagStatus](#) [ADC_DC_TrigStatus](#) (ADC_DC_Module_TypeDef ADC_DC_Module)
Проверка состояния выходного триггера компаратора.
- void [ADC_DC_TrigStatusClear](#) (ADC_DC_Module_TypeDef ADC_DC_Module)
Сброс выходного триггера цифрового компаратора.
- uint32_t [ADC_DC_GetLastData](#) (ADC_DC_Module_TypeDef ADC_DC_Module)
Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.

6.6.1 Подробное описание

6.6.2 Функции

6.6.2.1 void [ADC_Cmd](#) (ADC_Module_TypeDef ADC_Module, [FunctionalState](#) State)

Включение модуля АЦП.

Аргументы

ADC_Module	Выбор АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 108

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

6.6.2.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1017

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

6.6.2.3 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
--------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 1033

Перекрестные ссылки IS_ADC_DC_MODULE.

6.6.2.4 FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
--------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние триггера.
------------	-----------------------------

См. определение в файле niietcm4_adc.c строка 1051

Перекрестные ссылки IS_ADC_DC_MODULE.

6.6.2.5 void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Сброс выходного триггера цифрового компаратора.

Внимание

Одновременно со сбросом триггеров 0 и 1 компаратора сбрасываются триггеры 10 и 11 компаратора соответственно. То же самое справедливо и для обратного случая. Это происходит аппаратно и программными методами не обходится.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1079

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

6.6.2.6 void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
FunctionalState State)

Включение секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 848

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

6.6.2.7 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 976

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.8 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1001

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.9 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 936

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.10 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 961

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.11 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 898

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.12 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение результата измерений из буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 880

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.13 uint32_t ADC_SEQ_GetFIFOLoad (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 916

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.14 void ADC_SEQ_SWReq ()

Программный запуск измерений всех разрешенных секвенсоров.

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 868

6.7 Инициализация

Группы

- [Модули АЦП](#)
- [Цифровые компараторы](#)
- [Секвенсоры](#)

6.7.1 Подробное описание

6.8 Модули АЦП

Функции

- void [ADC_DeInit](#) ([ADC_Module_TypeDef](#) ADC_Module)
Устанавливает все регистры модуля АЦП значениями по умолчанию.
- void [ADC_Init](#) ([ADC_Module_TypeDef](#) ADC_Module, [ADC_Init_TypeDef](#) *ADC_InitStruct)
Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.
- void [ADC_StructInit](#) ([ADC_Init_TypeDef](#) *ADC_InitStruct)
Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

6.8.1 Подробное описание

6.8.2 Функции

6.8.2.1 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 123

Перекрестные ссылки [ADC_Module_0](#), [ADC_Module_1](#), [ADC_Module_10](#), [ADC_Module_11](#), [ADC_Module_2](#), [ADC_Module_3](#), [ADC_Module_4](#), [ADC_Module_5](#), [ADC_Module_6](#), [ADC_Module_7](#), [ADC_Module_8](#), [ADC_Module_9](#) и [IS_ADC_MODULE](#).

6.8.2.2 void ADC_Init (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *ADC_InitStruct)

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
ADC_InitStruct	Указатель на структуру типа ADC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 199

Перекрестные ссылки [ADC_Init_TypeDef::ADC_Average](#), [ADC_Init_TypeDef::ADC_Measure_A](#), [ADC_Init_TypeDef::ADC_Measure_B](#), [ADC_Init_TypeDef::ADC_Mode](#), [ADC_Module_0](#), [ADC_Module_1](#), [ADC_Module_10](#), [ADC_Module_11](#), [ADC_Module_2](#), [ADC_Module_3](#), [ADC_Module_4](#), [ADC_Module_5](#), [ADC_Module_6](#), [ADC_Module_7](#), [ADC_Module_8](#), [ADC_Module_9](#), [ADC_Init_TypeDef::ADC_Phase](#), [ADC_Init_TypeDef::ADC_Resolution](#), [IS_ADC_AVERAGE](#), [IS_ADC_MEASURE](#), [IS_ADC_MODE](#), [IS_ADC_MODULE](#), [IS_ADC_PHASE](#) и [IS_ADC_RESOLUTION](#).

6.8.2.3 void ADC_StructInit (ADC_Init_TypeDef *ADC_InitStruct)

Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

Аргументы

ADC_Init↔ Struct	Указатель на структуру типа ADC_Init_TypeDef , которую необходимо проинициализировать.
---------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_adc.c` строка 283

Перекрестные ссылки `ADC_Init_TypeDef::ADC_Average`, `ADC_Average_Disable`, `ADC_Init_↔
TypeDef::ADC_Measure_A`, `ADC_Init_TypeDef::ADC_Measure_B`, `ADC_Measure_Single`, `ADC↔
_Init_TypeDef::ADC_Mode`, `ADC_Mode_Powerdown`, `ADC_Init_TypeDef::ADC_Phase`, `ADC_↔
Init_TypeDef::ADC_Resolution` и `ADC_Resolution_12bit`.

6.9 Цифровые компараторы

Функции

- void [ADC_DC_DeInit](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.
- void [ADC_DC_Init](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [ADC_DC_Init_TypeDef](#) *ADC_DC_InitStruct)
Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры A↔DC_DC_InitStruct.
- void [ADC_DC_StructInit](#) ([ADC_DC_Init_TypeDef](#) *ADC_DC_InitStruct)
Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

6.9.1 Подробное описание

6.9.2 Функции

6.9.2.1 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 300

Перекрестные ссылки IS_ADC_DC_MODULE.

6.9.2.2 void ADC_DC_Init (ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры A↔DC_DC_InitStruct.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 319

Перекрестные ссылки [ADC_DC_Init_TypeDef::ADC_DC_Channel](#), [ADC_DC_Init_TypeDef::A↔DC_DC_Condition](#), [ADC_DC_Init_TypeDef::ADC_DC_Mode](#), [ADC_DC_Init_TypeDef::ADC_↔DC_ThresholdHigh](#), [ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow](#), [IS_ADC_DC](#), [IS_ADC_↔DC_CHANNEL](#), [IS_ADC_DC_CONDITION](#), [IS_ADC_DC_MODE](#) и [IS_ADC_DC_THRESHO↔LD](#).

6.9.2.3 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 342

Перекрестные ссылки [ADC_DC_Init_TypeDef::ADC_DC_Channel](#), [ADC_DC_Channel_None](#), [ADC_DC_Init_TypeDef::ADC_DC_Condition](#), [ADC_DC_Condition_Low](#), [ADC_DC_Init_TypeDef::ADC_DC_Mode](#), [ADC_DC_Mode_Single](#), [ADC_DC_Init_TypeDef::ADC_DC_↔ThresholdHigh](#) и [ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow](#).

6.10 Секвенсоры

Функции

- void [ADC_SEQ_DeInit](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Устанавливает все регистры выбранного секвенсора значениями по умолчанию.
- void [ADC_SEQ_Init](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [ADC_SEQ_Init_TypeDef](#) *ADC_SEQ_InitStruct)
Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.
- void [ADC_SEQ_StructInit](#) ([ADC_SEQ_Init_TypeDef](#) *ADC_SEQ_InitStruct)
Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

6.10.1 Подробное описание

6.10.2 Функции

6.10.2.1 void ADC_SEQ_DeInit (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

ADC_SEQ_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
---------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 358

Перекрестные ссылки ADC_SEQ_Module_0, ADC_SEQ_Module_1, ADC_SEQ_Module_2, ADC_↔
C_SEQ_Module_3, ADC_SEQ_Module_4, ADC_SEQ_Module_5, ADC_SEQ_Module_6, ADC_↔
_SEQ_Module_7 и IS_ADC_SEQ_MODULE.

6.10.2.2 void ADC_SEQ_Init (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 418

Перекрестные ссылки ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_SEQ_Init_TypeDef::AD_↔
C_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay, ADC_SEQ_↔
Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent, ADC_SEQ_↔
Init_TypeDef::ADC_SEQ_SWReqEn, IS_ADC_CHANNEL, IS_ADC_DC, IS_ADC_SEQ_CON_↔
VERSION_COUNT, IS_ADC_SEQ_CONVERSION_DELAY, IS_ADC_SEQ_MODULE, IS_AD_↔
C_SEQ_START_EVENT и IS_FUNCTIONAL_STATE.

6.10.2.3 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которую необходимо проинициализировать.
-------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 453

Перекрестные ссылки ADC_Channel_None, ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_DC_↔None, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::AD_↔C_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef_↔::ADC_SEQ_StartEvent, ADC_SEQ_StartEvent_SWReq и ADC_SEQ_Init_TypeDef::ADC_SE_↔Q_SWReqEn.

6.11 Конфигурация секвенсоров для DMA

Функции

- void [ADC_SEQ_DMAConfig](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [ADC_SEQ_FIFOLevel_TypeDef](#) ADC_SEQ_FIFOLevel)
Конфигурирует выбранный секвенсор для работы с DMA.
- void [ADC_SEQ_DMACmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)
Включает для выбранного секвенсора генерирование запросов DMA.
- [FlagStatus](#) [ADC_SEQ_DMAErrorStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.
- void [ADC_SEQ_DMAErrorStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Сброс статуса ошибки DMA.

6.11.1 Подробное описание

6.11.2 Функции

6.11.2.1 void [ADC_SEQ_DMACmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

ADC_SEQ_↔Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_adc.c` строка 489

Перекрестные ссылки `IS_ADC_SEQ_MODULE` и `IS_FUNCTIONAL_STATE`.

6.11.2.2 void [ADC_SEQ_DMAConfig](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [ADC_SEQ_FIFOLevel_TypeDef](#) ADC_SEQ_FIFOLevel)

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

ADC_SEQ_↔Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔Module_TypeDef .
ADC_SEQ_↔FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из ADC_SEQ_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_adc.c` строка 472

Перекрестные ссылки `IS_ADC_SEQ_FIFO_LEVEL` и `IS_ADC_SEQ_MODULE`.

6.11.2.3 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущие состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 505

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.11.2.4 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 530

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.12 Конфигурация прерываний

Группы

- [Цифровые компараторы](#)
- [Секвенсоры](#)

6.12.1 Подробное описание

6.13 Цифровые компараторы

Функции

- void [ADC_DC_ITCmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: [ADC_DC_ITGenCmd](#) и [ADC_DC_ITMaskCmd](#).
- void [ADC_DC_ITConfig](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [ADC_DC_Mode_TypeDef](#) ADC_DC_Mode, [ADC_DC_Condition_TypeDef](#) ADC_DC_Condition)
Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.
- void [ADC_DC_ITGenCmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
Разрешает компаратору генерировать сигнал прерывания.
- void [ADC_DC_ITMaskCmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
Маскирование сигнала прерывания цифрового компаратора.
- [FlagStatus](#) [ADC_DC_ITRawStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Проверка флагов немаскированных прерываний.
- [FlagStatus](#) [ADC_DC_ITMaskedStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Проверка флагов маскированных прерываний.
- void [ADC_DC_ITStatusClear](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.

6.13.1 Подробное описание

6.13.2 Функции

6.13.2.1 void [ADC_DC_ITCmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)

Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: [ADC_DC_ITGenCmd](#) и [ADC_DC_ITMaskCmd](#).

Аргументы

ADC_DC_Mode_TypeDef	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 589

Перекрестные ссылки [ADC_DC_ITGenCmd\(\)](#), [ADC_DC_ITMaskCmd\(\)](#), [IS_ADC_DC_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

```
6.13.2.2 void ADC_DC_ITConfig ( ADC_DC_Module_TypeDef ADC_DC_Module,  
ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef  
ADC_DC_Condition )
```

Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
ADC_DC_↔ Mode	Режим срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Mode_TypeDef .
ADC_DC_↔ Condition	Условие срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Condition_TypeDef .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 613

Перекрестные ссылки IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_MODULE.

6.13.2.3 void ADC_DC_ITGenCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 546

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

6.13.2.4 void ADC_DC_ITMaskCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 563

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

6.13.2.5 FlagStatus ADC_DC_ITMaskedStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов маскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из AD↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 655

Перекрестные ссылки IS_ADC_DC_MODULE.

6.13.2.6 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из AD↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 630

Перекрестные ссылки IS_ADC_DC_MODULE.

6.13.2.7 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из AD↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 681

Перекрестные ссылки IS_ADC_DC_MODULE.

6.14 Секвенсоры

Функции

- void [ADC_SEQ_ITCmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)
Включение прерывания секвенсора.
- void [ADC_SEQ_ITConfig](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, [FunctionalState](#) ADC_SEQ_ITCountSEQRst)
Настройка вызова прерывания секвенсора.
- uint32_t [ADC_SEQ_GetITCount](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.
- void [ADC_SEQ_ITCountRst](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Сброс счетчика прерываний секвенсора.
- [FlagStatus](#) [ADC_SEQ_ITRawStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка флагов немаскированных прерываний.
- [FlagStatus](#) [ADC_SEQ_ITMaskedStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка флагов маскированных прерываний.
- void [ADC_SEQ_ITStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.

6.14.1 Подробное описание

6.14.2 Функции

6.14.2.1 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

ITCount

См. определение в файле niietcm4_adc.c строка 750

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.2 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 697

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

6.14.2.3 void ADC_SEQ_ITConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)

Настройка вызова прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ ITRate	Значение количества перезапусков модулей АЦП секвенсором после которого генерируется прерывание. Параметр принимает любое значение из диапазона 1 - 256.
ADC_SEQ_↔ ITCountSEQRst	Разрешение сброса счетчика прерываний по запуску секвенсора. Если запретить, то счетчик можно будет сбрасывать только программно через ADC_SEQ_IT↔ CountRst . Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 725

Перекрестные ссылки IS_ADC_SEQ_IT_RATE, IS_ADC_SEQ_MODULE и IS_FUNCTIONAL↔
L_STATE.

6.14.2.4 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 768

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.5 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

--

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 807

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.6 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 782

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.7 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 832

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.15 Константы

Группы

- [Основная область флеш](#)
- [Информационная область флеш](#)

Макросы

- `#define BOOTFLASH_MAGIC_KEY ((uint32_t)0xA4420000)`
Ключ для проведения операций с контроллером загрузочной флеш.

6.15.1 Подробное описание

6.16 Основная область флеш

Макросы

- `#define BOOTFLASH_PAGE_SIZE_BYTES ((uint32_t)8192)`
- `#define BOOTFLASH_PAGE_TOTAL ((uint32_t)128)`
- `#define BOOTFLASH_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BOOTFLASH_PAGE_TOTAL)`
- `#define IS_BOOTFLASH_PAGE_NUM(PAGE_NUM) (PAGE_NUM < BOOTFLASH_PAGE_TOTAL)`

Макрос проверки номера страницы основной области загрузочной флеш на попадание в допустимый диапазон.

6.16.1 Подробное описание

6.16.2 Макросы

6.16.2.1 `#define BOOTFLASH_PAGE_SIZE_BYTES ((uint32_t)8192)`

Размер страницы в байтах.

См. определение в файле `niietcm4_bootflash.h` строка 62

Используется в `BOOTFLASH_Info_PageErase()` и `BOOTFLASH_PageErase()`.

6.16.2.2 `#define BOOTFLASH_PAGE_TOTAL ((uint32_t)128)`

Общее количество страниц.

См. определение в файле `niietcm4_bootflash.h` строка 63

6.16.2.3 `#define BOOTFLASH_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BOOTFLASH_PAGE_TOTAL)`

Общий размер основной области.

См. определение в файле `niietcm4_bootflash.h` строка 64

6.17 Информационная область флеш

Макросы

- `#define BOOTFLASH_INFO_PAGE_SIZE_BYTES BOOTFLASH_PAGE_SIZE_BYTES`
- `#define BOOTFLASH_INFO_PAGE_TOTAL ((uint32_t)1)`
- `#define BOOTFLASH_INFO_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BO←
OTFLASH_PAGE_TOTAL)`
- `#define IS_BOOTFLASH_INFO_PAGE_NUM(PAGE_NUM) (PAGE_NUM < BOOTFLA←
SH_INFO_PAGE_TOTAL)`

Макрос проверки номера страницы информационной области загрузочной флеш на попадание в допустимый диапазон.

6.17.1 Подробное описание

6.17.2 Макросы

6.17.2.1 `#define BOOTFLASH_INFO_PAGE_SIZE_BYTES BOOTFLASH_PAGE_SIZE_BY← TES`

Размер страницы в байтах.

См. определение в файле `niietcm4_bootflash.h` строка 80

6.17.2.2 `#define BOOTFLASH_INFO_PAGE_TOTAL ((uint32_t)1)`

Общее количество страниц.

См. определение в файле `niietcm4_bootflash.h` строка 81

6.17.2.3 `#define BOOTFLASH_INFO_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTE← S*BOOTFLASH_PAGE_TOTAL)`

Общий размер информационной области.

См. определение в файле `niietcm4_bootflash.h` строка 82

6.18 Типы

Макросы

- `#define IS_BOOTFLASH_STATUS(STATUS)`
Макрос проверки аргументов типа `BOOTFLASH_Status_TypeDef`.

Перечисления

- `enum BOOTFLASH_Status_TypeDef { BOOTFLASH_Status_None = ((uint32_t)0), BOOTFLASH_Status_Complete = ((uint32_t)1), BOOTFLASH_Status_Error = ((uint32_t)3) }`
Статус работы контроллера загрузочной флеш-памяти.

6.18.1 Подробное описание

6.18.2 Макросы

6.18.2.1 `#define IS_BOOTFLASH_STATUS(STATUS)`

Макроопределение:

```
((STATUS) == BOOTFLASH_Status_None) || \
((STATUS) == BOOTFLASH_Status_Complete) || \
((STATUS) == BOOTFLASH_Status_Error))
```

Макрос проверки аргументов типа `BOOTFLASH_Status_TypeDef`.

См. определение в файле `niietcm4_bootflash.h` строка 117

6.18.3 Перечисления

6.18.3.1 `enum BOOTFLASH_Status_TypeDef`

Статус работы контроллера загрузочной флеш-памяти.

Элементы перечислений

`BOOTFLASH_Status_None` Операция выполняется или отсутствует.

`BOOTFLASH_Status_Complete` Операция успешно завершена.

`BOOTFLASH_Status_Error` Операция завершена с ошибкой.

См. определение в файле `niietcm4_bootflash.h` строка 106

6.19 Функции

Группы

- [Основная область флеш](#)
- [Информационная область флеш](#)

Функции

- void [BOOTFLASH_Init](#) (uint32_t SysClkFreq)
Инициализирует тайминги доступа для контроллера загрузочной флеш.
- [BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus](#) ()
Статус работы контроллера загрузочной флэш.
- void [BOOTFLASH_OperationStatusClear](#) ()
Очищает статус работы контроллера загрузочной флэш.
- void [BOOTFLASH_ITCmd](#) ([FunctionalState](#) State)
Включение прерывания по завершению чтения/записи/стирания.

6.19.1 Подробное описание

6.19.2 Функции

6.19.2.1 void [BOOTFLASH_Init](#) (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера загрузочной флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
----------------------------	---------------------------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 75

6.19.2.2 void [BOOTFLASH_ITCmd](#) ([FunctionalState](#) State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-----------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.19.2.3 [BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus](#) ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из BOOTFLASH_H_Status_TypeDef .
--------	---

См. определение в файле niietcm4_bootflash.c строка 88

6.19.2.4 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 102

6.20 Основная область флеш

Функции

- void **BOOTFLASH_Write** (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.
- void **BOOTFLASH_PageErase** (uint32_t PageNum)
Стирание указанной страницы основной области загрузочной флеш.
- void **BOOTFLASH_FullErase** ()
Полная очистка основной области загрузочной флеш.

6.20.1 Подробное описание

6.20.2 Функции

6.20.2.1 void BOOTFLASH_FullErase ()

Полная очистка основной области загрузочной флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 112

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.20.2.2 void BOOTFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области загрузочной флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 144

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_↔ BOOTFLASH_PAGE_NUM.

6.20.2.3 void BOOTFLASH_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.

Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 128

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.21 Информационная область флеш

Функции

- void [BOOTFLASH_Info_Write](#) (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.
- void [BOOTFLASH_Info_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы информационной области загрузочной флеш.

6.21.1 Подробное описание

6.21.2 Функции

6.21.2.1 void BOOTFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области загрузочной флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 179

Перекрестные ссылки [BOOTFLASH_MAGIC_KEY](#), [BOOTFLASH_PAGE_SIZE_BYTES](#) и [IS_BOOTFLASH_INFO_PAGE_NUM](#).

6.21.2.2 void BOOTFLASH_Info_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 163

Перекрестные ссылки [BOOTFLASH_MAGIC_KEY](#).

6.22 Типы

Структуры данных

- struct `CAP_Init_TypeDef`
Структура инициализации блока захвата в целом.
- struct `CAP_Capture_Init_TypeDef`
Структура инициализации режима захвата.
- struct `CAP_PWM_Init_TypeDef`
Структура инициализации режима ШИМ.

Макросы

- `#define IS_CAP_CAPTURE_POLARITY(CAPTURE_POLARITY)`
Макрос проверки аргументов типа `CAP_Capture_Polarity_TypeDef`.
- `#define IS_CAP_HALT(HALT)`
Макрос проверки аргументов типа `CAP_Halt_TypeDef`.
- `#define IS_CAP_SYNC_OUT(SYNC_OUT)`
Макрос проверки аргументов типа `CAP_SyncOut_TypeDef`.
- `#define IS_CAP_CAPTURE_MODE(CAPTURE_MODE)`
Макрос проверки аргументов типа `CAP_Capture_Mode_TypeDef`.
- `#define IS_CAP_PWM_POLARITY(PWM_POLARITY)`
Макрос проверки аргументов типа `CAP_PWM_Polarity_TypeDef`.
- `#define IS_CAP_MODE(MODE)`
Макрос проверки аргументов типа `CAP_Mode_TypeDef`.
- `#define IS_CAP_CAPTURE_PRESCALE(PRESCALE) ((PRESCALE) < ((uint32_t)0x40))`
Проверка значения предварительного делителя событий на попадание в допустимый диапазон.
- `#define IS_CAP_CAPTURE_STOP_VAL(STOP_VAL) ((STOP_VAL) < ((uint32_t)4))`
Проверка значения счетчика событий для остановки одиночного режима захвата на попадание в допустимый диапазон.

Перечисления

- enum `CAP_Capture_Polarity_TypeDef` { `CAP_Capture_Polarity_PosEdge`, `CAP_Capture_Polarity_NegEdge` }
Выбор фронта захвата.
- enum `CAP_Halt_TypeDef` { `CAP_Halt_Stop`, `CAP_Halt_StopOnZero`, `CAP_Halt_Free` }
Выбор режима остановки таймера при отладке.
- enum `CAP_SyncOut_TypeDef` { `CAP_SyncOut_Bypass`, `CAP_SyncOut_TimerEqPeriod`, `CAP_SyncOut_Disable` }
Выбор источника выходного сигнала синхронизации.
- enum `CAP_Capture_Mode_TypeDef` { `CAP_Capture_Mode_Cycle`, `CAP_Capture_Mode_Single` }
Выбор режима работы захвата.
- enum `CAP_PWM_Polarity_TypeDef` { `CAP_PWM_Polarity_Pos`, `CAP_PWM_Polarity_Neg` }
Выбор активного уровня в режиме ШИМ.
- enum `CAP_Mode_TypeDef` { `CAP_Mode_Capture`, `CAP_Mode_PWM` }
Выбор режима работы блока захвата.

6.22.1 Подробное описание

6.22.2 Макросы

6.22.2.1 #define IS_CAP_CAPTURE_MODE(CAPTURE_MODE)

Макроопределение:

```
((CAPTURE_MODE) == CAP_Capture_Mode_Single) || \
  ((CAPTURE_MODE) == CAP_Capture_Mode_Cycle))
```

Макрос проверки аргументов типа `CAP_Capture_Mode_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 121

Используется в `CAP_Capture_Init()`.

6.22.2.2 #define IS_CAP_CAPTURE_POLARITY(CAPTURE_POLARITY)

Макроопределение:

```
((CAPTURE_POLARITY) == CAP_Capture_Polarity_PosEdge) || \
  ((CAPTURE_POLARITY) == CAP_Capture_Polarity_NegEdge))
```

Макрос проверки аргументов типа `CAP_Capture_Polarity_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 66

Используется в `CAP_Capture_Init()`.

6.22.2.3 #define IS_CAP_HALT(HALT)

Макроопределение:

```
((HALT) == CAP_Halt_Stop) || \
  ((HALT) == CAP_Halt_StopOnZero) || \
  ((HALT) == CAP_Halt_Free))
```

Макрос проверки аргументов типа `CAP_Halt_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 84

Используется в `CAP_Init()`.

6.22.2.4 #define IS_CAP_MODE(MODE)

Макроопределение:

```
((MODE) == CAP_Mode_Capture) || \
  ((MODE) == CAP_Mode_PWM))
```

Макрос проверки аргументов типа `CAP_Mode_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 155

Используется в `CAP_Init()`.

6.22.2.5 `#define IS_CAP_PWM_POLARITY(PWM_POLARITY)`

Макроопределение:

```
((PWM_POLARITY) == CAP_PWM_Polarity_Pos) || \
  ((PWM_POLARITY) == CAP_PWM_Polarity_Neg))
```

Макрос проверки аргументов типа `CAP_PWM_Polarity_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 138

Используется в `CAP_PWM_Init()`.

6.22.2.6 `#define IS_CAP_SYNC_OUT(SYNC_OUT)`

Макроопределение:

```
((SYNC_OUT) == CAP_SyncOut_Bypass) || \
  ((SYNC_OUT) == CAP_SyncOut_TimerEqPeriod) || \
  ((SYNC_OUT) == CAP_SyncOut_Disable))
```

Макрос проверки аргументов типа `CAP_SyncOut_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 103

Используется в `CAP_Init()`.

6.22.3 Перечисления

6.22.3.1 `enum CAP_Capture_Mode_TypeDef`

Выбор режима работы захвата.

Элементы перечислений

`CAP_Capture_Mode_Cycle` Циклический захват.

`CAP_Capture_Mode_Single` Однократный захват.

См. определение в файле `niietcm4_cap.h` строка 111

6.22.3.2 `enum CAP_Capture_Polarity_TypeDef`

Выбор фронта захвата.

Элементы перечислений

`CAP_Capture_Polarity_PosEdge` Захват по переднему фронту.

`CAP_Capture_Polarity_NegEdge` Захват по заднему фронту.

См. определение в файле `niietcm4_cap.h` строка 56

6.22.3.3 `enum CAP_Halt_TypeDef`

Выбор режима остановки таймера при отладке.

Элементы перечислений

`CAP_Halt_Stop` Мгновенная остановка таймера при отладке.

`CAP_Halt_StopOnZero` Остановка таймера при переполнении или сбросе (событие достижения 0).

`CAP_Halt_Free` Нормальный режим.

См. определение в файле `niietcm4_cap.h` строка 73

6.22.3.4 `enum CAP_Mode_TypeDef`

Выбор режима работы блока захвата.

Элементы перечислений

`CAP_Mode_Capture` Режим захвата.

`CAP_Mode_PWM` Режим ШИМ.

См. определение в файле `niietcm4_cap.h` строка 145

6.22.3.5 `enum CAP_PWM_Polarity_TypeDef`

Выбор активного уровня в режиме ШИМ.

Элементы перечислений

`CAP_PWM_Polarity_Pos` Высокий уровень является активным.

`CAP_PWM_Polarity_Neg` Низкий уровень является активным.

См. определение в файле `niietcm4_cap.h` строка 128

6.22.3.6 `enum CAP_SyncOut_TypeDef`

Выбор источника выходного сигнала синхронизации.

Элементы перечислений

`CAP_SyncOut_Bypass` Пропуск синхросигнала со входа на выход.

`CAP_SyncOut_TimerEqPeriod` Передача события равенства таймера и значения периода в качестве выходного сигнала синхронизации.

`CAP_SyncOut_Disable` Выходной сигнал синхронизации запрещен.

См. определение в файле `niietcm4_cap.h` строка 92

6.23 Константы

Группы

- [Маски источников прерываний](#)

6.23.1 Подробное описание

6.24 Маски источников прерываний

Макросы

- `#define CAP_ITSource_GeneralInt ((uint32_t)0x01)`
- `#define CAP_ITSource_CapEvent0 ((uint32_t)0x02)`
- `#define CAP_ITSource_CapEvent1 ((uint32_t)0x04)`
- `#define CAP_ITSource_CapEvent2 ((uint32_t)0x08)`
- `#define CAP_ITSource_CapEvent3 ((uint32_t)0x10)`
- `#define CAP_ITSource_TimerOvf ((uint32_t)0x20)`
- `#define CAP_ITSource_TimerEqPeriod ((uint32_t)0x40)`
- `#define CAP_ITSource_TimerEqCompare ((uint32_t)0x80)`
- `#define CAP_ITSource_All ((uint32_t)0xFF)`
- `#define IS_CAP_IT_SOURCE(IT_SOURCE) (((IT_SOURCE) != (uint32_t)0x0000) && (((IT_SOURCE) & (uint32_t)0xFFFFFFF0) == ((uint32_t)0x00)))`

Макрос проверки источников прерываний на попадание в допустимый диапазон.

- `#define IS_CAP_IT_SOURCE_SINGLE(IT_SOURCE)`

Макрос проверки источника прерываний при работе с ними по отдельности (опрос флагов).

6.24.1 Подробное описание

6.24.2 Макросы

6.24.2.1 `#define CAP_ITSource_All ((uint32_t)0xFF)`

Все источники выбраны.

См. определение в файле `niietcm4_cap.h` строка 252

6.24.2.2 `#define CAP_ITSource_CapEvent0 ((uint32_t)0x02)`

Событие захвата 0.

См. определение в файле `niietcm4_cap.h` строка 245

6.24.2.3 `#define CAP_ITSource_CapEvent1 ((uint32_t)0x04)`

Событие захвата 1.

См. определение в файле `niietcm4_cap.h` строка 246

6.24.2.4 `#define CAP_ITSource_CapEvent2 ((uint32_t)0x08)`

Событие захвата 2.

См. определение в файле `niietcm4_cap.h` строка 247

6.24.2.5 `#define CAP_ITSource_CapEvent3 ((uint32_t)0x10)`

Событие захвата 3.

См. определение в файле `niietcm4_cap.h` строка 248

6.24.2.6 `#define CAP_ITSource_GeneralInt ((uint32_t)0x01)`

Общее прерывание.

См. определение в файле `niietcm4_cap.h` строка 244

6.24.2.7 `#define CAP_ITSource_TimerEqCompare ((uint32_t)0x80)`

Счетчик таймера равен значению сравнения (в режиме ШИМ).

См. определение в файле `niietcm4_cap.h` строка 251

6.24.2.8 `#define CAP_ITSource_TimerEqPeriod ((uint32_t)0x40)`

Счетчик таймера равен периоду (в режиме ШИМ).

См. определение в файле `niietcm4_cap.h` строка 250

6.24.2.9 `#define CAP_ITSource_TimerOvf ((uint32_t)0x20)`

Переполнение счетчика таймера.

См. определение в файле `niietcm4_cap.h` строка 249

6.24.2.10 `#define IS_CAP_IT_SOURCE_SINGLE(IT_SOURCE)`

Макроопределение:

```
((IT_SOURCE) == CAP_ITSource_GeneralInt) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent0) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent1) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent2) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent3) || \
  ((IT_SOURCE) == CAP_ITSource_TimerOvf) || \
  ((IT_SOURCE) == CAP_ITSource_TimerEqPeriod) || \
  ((IT_SOURCE) == CAP_ITSource_TimerEqCompare))
```

Макрос проверки источника прерываний при работе с ними по отдельности (опрос флагов).

См. определение в файле `niietcm4_cap.h` строка 265

Используется в `CAP_ITStatus()`.

6.25 Функции

Группы

- [Конфигурация](#)
- [Режим ШИМ](#)
- [Режим захвата](#)
- [Прерывания](#)

6.25.1 Подробное описание

6.26 Конфигурация

Функции

- void [CAP_DeInit](#) (NT_CAP_TypeDef *CAPx)
Устанавливает все регистры блока захвата значениями по умолчанию.
- void [CAP_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_Init_TypeDef](#) *CAP_InitStruct)
Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.
- void [CAP_StructInit](#) ([CAP_Init_TypeDef](#) *CAP_InitStruct)
Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.
- void [CAP_TimerCmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение работы таймера, выбранного блока захвата.
- void [CAP_SetTimer](#) (NT_CAP_TypeDef *CAPx, uint32_t TimerVal)
Установка текущего значения счетчика напрямую.
- void [CAP_SetShadowTimer](#) (NT_CAP_TypeDef *CAPx, uint32_t TimerVal)
Установка теневого значения таймера для отложенной записи.
- uint32_t [CAP_GetTimer](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения таймера.
- uint32_t [CAP_GetShadowTimer](#) (NT_CAP_TypeDef *CAPx)
Получение отложенного значения таймера.
- void [CAP_SyncCmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение синхронизации.
- void [CAP_SwSync](#) (NT_CAP_TypeDef *CAPx)
Проведение программной синхронизации.

6.26.1 Подробное описание

6.26.2 Функции

6.26.2.1 void CAP_DeInit (NT_CAP_TypeDef * CAPx)

Устанавливает все регистры блока захвата значениями по умолчанию.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 68

Перекрестные ссылки IS_CAP_ALL_PERIPH, RCC_PeriphRst_CAP0, RCC_PeriphRst_CAP1, RCC_PeriphRst_CAP2, RCC_PeriphRst_CAP3, RCC_PeriphRst_CAP4, RCC_PeriphRst_CAP5 и RCC_PeriphRstCmd().

6.26.2.2 uint32_t CAP_GetShadowTimer (NT_CAP_TypeDef * CAPx)

Получение отложенного значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 218

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.26.2.3 uint32_t CAP_GetTimer (NT_CAP_TypeDef * CAPx)

Получение текущего значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 205

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.26.2.4 void CAP_Init (NT_CAP_TypeDef * CAPx, CAP_Init_TypeDef * CAP_InitStruct)

Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef, которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 111

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Init_TypeDef::CAP_Mode, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut, IS_CAP_ALL_PERIPH, IS_CAP_HALT, IS_CAP_MODE и IS_CAP_SYNC_OUT.

6.26.2.5 void CAP_SetShadowTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)

Установка теневого значения таймера для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 192

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.26.2.6 void CAP_SetTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)

Установка текущего значения счетчика напрямую.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 178

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.26.2.7 void CAP_StructInit (CAP_Init_TypeDef * CAP_InitStruct)

Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.

Аргументы

CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef, которую необходимо проинициализировать.
----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 147

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Halt_Stop, CAP_Init_TypeDef::CAP_P_Mode, CAP_Mode_Capture, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut и CAP_SyncOut_Bypass.

6.26.2.8 void CAP_SwSync (NT_CAP_TypeDef * CAPx)

Проведение программной синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 231

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.26.2.9 void CAP_SyncCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
State	Выбор состояния. Параметр принимает любое значение из FunctionalState.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 132

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

6.26.2.10 void CAP_TimerCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение работы таймера, выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 163

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

6.27 Режим ШИМ

Функции

- void [CAP_PWM_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_PWM_Init_TypeDef](#) *CAP_PWM_InitStruct)

Инициализирует режим ШИМ блока CAPx согласно параметрам структуры CAP_PWM_InitStruct.
- void [CAP_PWM_StructInit](#) ([CAP_PWM_Init_TypeDef](#) *CAP_PWM_InitStruct)

Заполнение каждого члена структуры CAP_PWM_InitStruct значениями по умолчанию.
- void [CAP_PWM_SetPeriod](#) (NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ.
- void [CAP_PWM_SetCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ.
- void [CAP_PWM_SetShadowPeriod](#) (NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ для отложенной записи.
- void [CAP_PWM_SetShadowCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ для отложенной записи.
- uint32_t [CAP_PWM_GetPeriod](#) (NT_CAP_TypeDef *CAPx)

Получение текущего периода ШИМ.
- uint32_t [CAP_PWM_GetCompare](#) (NT_CAP_TypeDef *CAPx)

Получение текущего значения сравнения ШИМ.
- uint32_t [CAP_PWM_GetShadowPeriod](#) (NT_CAP_TypeDef *CAPx)

Получение отложенного значения периода ШИМ.
- uint32_t [CAP_PWM_GetShadowCompare](#) (NT_CAP_TypeDef *CAPx)

Получение отложенного значения сравнения ШИМ.

6.27.1 Подробное описание

6.27.2 Функции

6.27.2.1 uint32_t CAP_PWM_GetCompare (NT_CAP_TypeDef * CAPx)

Получение текущего значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле niietcm4_cap.c строка 345

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.27.2.2 uint32_t CAP_PWM_GetPeriod (NT_CAP_TypeDef * CAPx)

Получение текущего периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле niietcm4_cap.c строка 332

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.27.2.3 uint32_t CAP_PWM_GetShadowCompare (NT_CAP_TypeDef * CAPx)

Получение отложенного значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле niietcm4_cap.c строка 371

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.27.2.4 uint32_t CAP_PWM_GetShadowPeriod (NT_CAP_TypeDef * CAPx)

Получение отложенного значения периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле niietcm4_cap.c строка 358

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.27.2.5 void CAP_PWM_Init (NT_CAP_TypeDef * CAPx, CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)

Инициализирует режим ШИМ блока CAPx согласно параметрам структуры CAP_PWM_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_PWM_InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef, которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 246

Перекрестные ссылки CAP_PWM_Init_TypeDef::CAP_PWM_Compare, CAP_PWM_Init_TypeDef::CAP_PWM_Period, CAP_PWM_Init_TypeDef::CAP_PWM_Polarity, CAP_PWM_SetCompare(), CAP_PWM_SetPeriod(), IS_CAP_ALL_PERIPH и IS_CAP_PWM_POLARITY.

6.27.2.6 void CAP_PWM_SetCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 291

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

6.27.2.7 void CAP_PWM_SetPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 277

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

6.27.2.8 void CAP_PWM_SetShadowCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 319

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.27.2.9 void CAP_PWM_SetShadowPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_cap.c` строка 305

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.27.2.10 `void CAP_PWM_StructInit (CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)`

Заполнение каждого члена структуры `CAP_PWM_InitStruct` значениями по умолчанию.

Аргументы

<code>CAP_PWM_↔ _InitStruct</code>	Указатель на структуру типа CAP_PWM_Init_TypeDef , которую необходимо проинициализировать.
--	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_cap.c` строка 263

Перекрестные ссылки `CAP_PWM_Init_TypeDef::CAP_PWM_Compare`, `CAP_PWM_Init_↔
TypeDef::CAP_PWM_Period`, `CAP_PWM_Init_TypeDef::CAP_PWM_Polarity` и `CAP_PWM_↔
Polarity_Pos`.

6.28 Режим захвата

Функции

- void [CAP_Capture_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_Capture_Init_TypeDef](#) *CAP_Capture_InitStruct)
Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.
- void [CAP_Capture_StructInit](#) ([CAP_Capture_Init_TypeDef](#) *CAP_Capture_InitStruct)
Заполнение каждого члена структуры CAP_Capture_InitStruct значениями по умолчанию.
- void [CAP_Capture_Cmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение захвата для выбранного блока захвата.
- void [CAP_Capture_SetCap0](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 0.
- void [CAP_Capture_SetCap1](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 1.
- void [CAP_Capture_SetCap2](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 2.
- void [CAP_Capture_SetCap3](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 3.
- uint32_t [CAP_Capture_GetCap0](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 0.
- uint32_t [CAP_Capture_GetCap1](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 1.
- uint32_t [CAP_Capture_GetCap2](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 2.
- uint32_t [CAP_Capture_GetCap3](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 3.

6.28.1 Подробное описание

6.28.2 Функции

6.28.2.1 void CAP_Capture_Cmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение захвата для выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 444

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

6.28.2.2 uint32_t CAP_Capture_GetCap0 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 0.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 519

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.28.2.3 uint32_t CAP_Capture_GetCap1 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 1.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 532

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.28.2.4 uint32_t CAP_Capture_GetCap2 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 2.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 545

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.28.2.5 uint32_t CAP_Capture_GetCap3 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 3.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 558

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.28.2.6 void CAP_Capture_Init (NT_CAP_TypeDef * CAPx, CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)

Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_↔ Capture_Init↔ Struct	Указатель на структуру типа CAP_Capture_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 386

Перекрестные ссылки `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`, `CAP_↔
Capture_Init_TypeDef::CAP_Capture_PolarityEvent1`, `CAP_Capture_Init_TypeDef::CAP_↔
Capture_PolarityEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`, `CAP_↔
Capture_Init_TypeDef::CAP_Capture_Prescale`, `CAP_Capture_Init_TypeDef::CAP_Capture_↔
RstEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1`, `CAP_Capture_Init_Type↔
Def::CAP_Capture_RstEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3`, `CAP_↔
Capture_Init_TypeDef::CAP_Capture_StopVal`, `CAP_Capture_Init_TypeDef::CAP_CaptureMode`,
`IS_CAP_ALL_PERIPH`, `IS_CAP_CAPTURE_MODE`, `IS_CAP_CAPTURE_POLARITY`, `IS_↔
CAP_CAPTURE_PRESCALE`, `IS_CAP_CAPTURE_STOP_VAL` и `IS_FUNCTIONAL_STATE`.

6.28.2.7 `void CAP_Capture_SetCap0 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 0.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 464

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.28.2.8 `void CAP_Capture_SetCap1 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 1.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 478

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.28.2.9 `void CAP_Capture_SetCap2 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 2.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 492

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.28.2.10 `void CAP_Capture_SetCap3 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 3.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 506

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.28.2.11 `void CAP_Capture_StructInit (CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)`

Заполнение каждого члена структуры `CAP_Capture_InitStruct` значениями по умолчанию.

Аргументы

<code>CAP_Capture_InitStruct</code>	Указатель на структуру типа <code>CAP_Capture_Init_TypeDef</code> , которую необходимо проинициализировать.
-------------------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 421

Перекрестные ссылки `CAP_Capture_Mode_Single`, `CAP_Capture_Polarity_PosEdge`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`, `CAP_Capture_Init_TypeDef::CAP_Capture_Prescale`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3`, `CAP_Capture_Init_TypeDef::CAP_Capture_StopVal` и `CAP_Capture_Init_TypeDef::CAP_CaptureMode`.

6.29 Прерывания

Функции

- void [CAP_ITCmd](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, [FunctionalState](#) State)
Разрешение работы прерывания выбранного блока захвата.
- void [CAP_ITForceCmd](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)
Принудительный вызов прерывания выбранного блока захвата.
- [FlagStatus](#) [CAP_ITStatus](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)
Чтение статуса флага источника прерывания выбранного блока захвата.
- void [CAP_ITStatusClear](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)
Сброс флагов источников прерываний выбранного блока захвата.
- [FlagStatus](#) [CAP_ITPendStatus](#) (NT_CAP_TypeDef *CAPx)
Чтение статуса прерывания выбранного блока захвата.
- void [CAP_ITPendClear](#) (NT_CAP_TypeDef *CAPx)
Сброс флага прерывания выбранного блока захвата.

6.29.1 Подробное описание

6.29.2 Функции

6.29.2.1 void [CAP_ITCmd](#) (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource, [FunctionalState](#) State)

Разрешение работы прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 575

Перекрестные ссылки IS_CAP_ALL_PERIPH, IS_CAP_IT_SOURCE и IS_FUNCTIONAL_STATE.

6.29.2.2 void [CAP_ITForceCmd](#) (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Принудительный вызов прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 599

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

6.29.2.3 void CAP_ITPendClear (NT_CAP_TypeDef * CAPx)

Сброс флага прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 680

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.29.2.4 FlagStatus CAP_ITPendStatus (NT_CAP_TypeDef * CAPx)

Чтение статуса прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 656

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.29.2.5 FlagStatus CAP_ITStatus (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Чтение статуса флага источника прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT← Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 615

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE_SINGLE.

6.29.2.6 void CAP_ITStatusClear (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Сброс флагов источников прерываний выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 642

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

6.30 Константы

Группы

- [Маски для CHANNEL_CFG](#)
Битовые позиции и маски регистра CHANNEL_CFG в [DMA_Channel_TypeDef](#).
- [Маски каналов DMA](#)

6.30.1 Подробное описание

6.31 Маски для CHANNEL_CFG

Битовые позиции и маски регистра CHANNEL_CFG в [DMA_Channel_TypeDef](#).

Макросы

- `#define CHANNEL_CFG_CYCLE_CTRL_Pos 0`
- `#define CHANNEL_CFG_NEXT_USEBURST_Pos 3`
- `#define CHANNEL_CFG_N_MINUS_1_Pos 4`
- `#define CHANNEL_CFG_R_POWER_Pos 14`
- `#define CHANNEL_CFG_SRC_PROT_CTRL_Pos 18`
- `#define CHANNEL_CFG_DST_PROT_CTRL_Pos 21`
- `#define CHANNEL_CFG_SRC_SIZE_Pos 24`
- `#define CHANNEL_CFG_SRC_INC_Pos 26`
- `#define CHANNEL_CFG_DST_SIZE_Pos 28`
- `#define CHANNEL_CFG_DST_INC_Pos 30`
- `#define CHANNEL_CFG_CYCLE_CTRL_Msk ((uint32_t)0x00000007)`
- `#define CHANNEL_CFG_NEXT_USEBURST_Msk ((uint32_t)0x00000008)`
- `#define CHANNEL_CFG_N_MINUS_1_Msk ((uint32_t)0x00003FF0)`
- `#define CHANNEL_CFG_R_POWER_Msk ((uint32_t)0x0003C000)`
- `#define CHANNEL_CFG_SRC_PROT_CTRL_Msk ((uint32_t)0x001C0000)`
- `#define CHANNEL_CFG_DST_PROT_CTRL_Msk ((uint32_t)0x00E00000)`
- `#define CHANNEL_CFG_SRC_SIZE_Msk ((uint32_t)0x03000000)`
- `#define CHANNEL_CFG_SRC_INC_Msk ((uint32_t)0x0C000000)`
- `#define CHANNEL_CFG_DST_SIZE_Msk ((uint32_t)0x30000000)`
- `#define CHANNEL_CFG_DST_INC_Msk ((uint32_t)0xC0000000)`

6.31.1 Подробное описание

Битовые позиции и маски регистра CHANNEL_CFG в [DMA_Channel_TypeDef](#).

6.31.2 Макросы

6.31.2.1 `#define CHANNEL_CFG_CYCLE_CTRL_Msk ((uint32_t)0x00000007)`

Поле задания типа цикла DMA

См. определение в файле `niietcm4_dma.h` строка 68

6.31.2.2 `#define CHANNEL_CFG_CYCLE_CTRL_Pos 0`

Поле задания типа цикла DMA

См. определение в файле `niietcm4_dma.h` строка 57

6.31.2.3 `#define CHANNEL_CFG_DST_INC_Msk ((uint32_t)0xC0000000)`

Шаг инкремента адреса приемника

См. определение в файле `niietcm4_dma.h` строка 77

6.31.2.4 `#define CHANNEL_CFG_DST_INC_Pos 30`

Шаг инкремента адреса приемника

См. определение в файле `niietcm4_dma.h` строка 66

6.31.2.5 `#define CHANNEL_CFG_DST_PROT_CTRL_Msk ((uint32_t)0x00E00000)`

Защита шины АHB-Lite при записи данных в приемник

См. определение в файле `niietcm4_dma.h` строка 73

6.31.2.6 `#define CHANNEL_CFG_DST_PROT_CTRL_Pos 21`

Защита шины АHB-Lite при записи данных в приемник

См. определение в файле `niietcm4_dma.h` строка 62

6.31.2.7 `#define CHANNEL_CFG_DST_SIZE_Msk ((uint32_t)0x30000000)`

Разрядность данных приемника

См. определение в файле `niietcm4_dma.h` строка 76

6.31.2.8 `#define CHANNEL_CFG_DST_SIZE_Pos 28`

Разрядность данных приемника

См. определение в файле `niietcm4_dma.h` строка 65

6.31.2.9 `#define CHANNEL_CFG_N_MINUS_1_Msk ((uint32_t)0x00003FF0)`

Общее количество передач в цикле работы

См. определение в файле `niietcm4_dma.h` строка 70

6.31.2.10 `#define CHANNEL_CFG_N_MINUS_1_Pos 4`

Общее количество передач в цикле работы

См. определение в файле `niietcm4_dma.h` строка 59

6.31.2.11 `#define CHANNEL_CFG_NEXT_USEBURST_Msk ((uint32_t)0x00000008)`

Контролирует установку соответствующего каналу бита в регистре `NT_DMA->CHNL_USEBURST_SET`

См. определение в файле `niietcm4_dma.h` строка 69

6.31.2.12 `#define CHANNEL_CFG_NEXT_USEBURST_Pos 3`

Контролирует установку соответствующего каналу бита в регистре `NT_DMA->CHNL_USEBURST_SET`

См. определение в файле `niietcm4_dma.h` строка 58

6.31.2.13 `#define CHANNEL_CFG_R_POWER_Msk ((uint32_t)0x0003C000)`

Количество передач до выполнения переарбитрации

См. определение в файле `niietcm4_dma.h` строка 71

6.31.2.14 `#define CHANNEL_CFG_R_POWER_Pos 14`

Количество передач до выполнения переарбитрации

См. определение в файле `niietcm4_dma.h` строка 60

6.31.2.15 `#define CHANNEL_CFG_SRC_INC_Msk ((uint32_t)0x0C000000)`

Шаг инкремента адреса источника

См. определение в файле `niietcm4_dma.h` строка 75

6.31.2.16 `#define CHANNEL_CFG_SRC_INC_Pos 26`

Шаг инкремента адреса источника

См. определение в файле `niietcm4_dma.h` строка 64

6.31.2.17 `#define CHANNEL_CFG_SRC_PROT_CTRL_Msk ((uint32_t)0x001C0000)`

Защита шины АHB-Lite при чтении данных из источника

См. определение в файле `niietcm4_dma.h` строка 72

6.31.2.18 `#define CHANNEL_CFG_SRC_PROT_CTRL_Pos 18`

Защита шины АHB-Lite при чтении данных из источника

См. определение в файле `niietcm4_dma.h` строка 61

6.31.2.19 `#define CHANNEL_CFG_SRC_SIZE_Msk ((uint32_t)0x03000000)`

Разрядность данных источника

См. определение в файле `niietcm4_dma.h` строка 74

6.31.2.20 `#define CHANNEL_CFG_SRC_SIZE_Pos 24`

Разрядность данных источника

См. определение в файле `niietcm4_dma.h` строка 63

6.32 Маски каналов DMA

Группы

- [Маски каналов по имени](#)
- [Маски каналов по номеру](#)

Макросы

- `#define DMA_Channel_All ((uint32_t)0x00FFFFFF)`
- `#define IS_DMA_CHANNEL(CHANNEL) (((CHANNEL) != (uint32_t)0x000000) && (((CHANNEL) & (uint32_t)0xFF000000) == ((uint32_t)0x0000)))`

Макрос проверки маски каналов на попадание в допустимый диапазон.

- `#define IS_GET_DMA_CHANNEL(CHANNEL)`

Макрос проверки маски канала при работе с каналами по отдельности.

6.32.1 Подробное описание

6.32.2 Макросы

6.32.2.1 `#define DMA_Channel_All ((uint32_t)0x00FFFFFF)`

Все каналы DMA

См. определение в файле `niietcm4_dma.h` строка 87

6.32.2.2 `#define IS_GET_DMA_CHANNEL(CHANNEL)`

Макроопределение:

```
(((CHANNEL) == (DMA_Channel_0 || DMA_Channel_UART0_TX)) || \
 DMA_Channel_UART1_TX)) || \
 ((CHANNEL) == (DMA_Channel_2 || \
 DMA_Channel_UART2_TX)) || \
 ((CHANNEL) == (DMA_Channel_3 || \
 DMA_Channel_UART3_TX)) || \
 ((CHANNEL) == (DMA_Channel_4 || \
 DMA_Channel_UART0_RX)) || \
 ((CHANNEL) == (DMA_Channel_5 || \
 DMA_Channel_UART1_RX)) || \
 ((CHANNEL) == (DMA_Channel_6 || \
 DMA_Channel_UART2_RX)) || \
 ((CHANNEL) == (DMA_Channel_7 || \
 DMA_Channel_UART3_RX)) || \
 ((CHANNEL) == (DMA_Channel_8 || \
 DMA_Channel_ADCSEQ0)) || \
 ((CHANNEL) == (DMA_Channel_9 || \
 DMA_Channel_ADCSEQ1)) || \
 ((CHANNEL) == (DMA_Channel_10 || \
 DMA_Channel_ADCSEQ2)) || \
 ((CHANNEL) == (DMA_Channel_11 || \
 DMA_Channel_ADCSEQ3)) || \
 ((CHANNEL) == (DMA_Channel_12 || \
 DMA_Channel_ADCSEQ4)) || \
 ((CHANNEL) == (DMA_Channel_13 || \
 DMA_Channel_ADCSEQ5)) || \
 ((CHANNEL) == (DMA_Channel_14 || \
 DMA_Channel_ADCSEQ6)) || \
 ((CHANNEL) == (DMA_Channel_15 || \
 DMA_Channel_ADCSEQ7)) || \
 ((CHANNEL) == (DMA_Channel_16 || \
 DMA_Channel_SPI0_TX)) || \
 ((CHANNEL) == (DMA_Channel_17 || \
 DMA_Channel_SPI1_TX)) || \
 ((CHANNEL) == (DMA_Channel_18 || \
 DMA_Channel_SPI2_TX)) || \
 ((CHANNEL) == (DMA_Channel_10 ||
```

```
DMA_Channel_SPI3_TX)) || \
((CHANNEL) == (DMA_Channel_20 ||
DMA_Channel_SPI0_RX)) || \
((CHANNEL) == (DMA_Channel_21 ||
DMA_Channel_SPI1_RX)) || \
((CHANNEL) == (DMA_Channel_22 ||
DMA_Channel_SPI2_RX)) || \
((CHANNEL) == (DMA_Channel_23 ||
DMA_Channel_SPI3_RX)))
```

Макрос проверки маски канала при работе с каналами по отдельности.

См. определение в файле niietcm4_dma.h строка 166

Используется в DMA_WaitOnReqStatus().

6.33 Маски каналов по имени

Макросы

- `#define DMA_Channel_UART0_TX ((uint32_t)0x00000001)`
- `#define DMA_Channel_UART1_TX ((uint32_t)0x00000002)`
- `#define DMA_Channel_UART2_TX ((uint32_t)0x00000004)`
- `#define DMA_Channel_UART3_TX ((uint32_t)0x00000008)`
- `#define DMA_Channel_UART0_RX ((uint32_t)0x00000010)`
- `#define DMA_Channel_UART1_RX ((uint32_t)0x00000020)`
- `#define DMA_Channel_UART2_RX ((uint32_t)0x00000040)`
- `#define DMA_Channel_UART3_RX ((uint32_t)0x00000080)`
- `#define DMA_Channel_ADCSEQ0 ((uint32_t)0x00000100)`
- `#define DMA_Channel_ADCSEQ1 ((uint32_t)0x00000200)`
- `#define DMA_Channel_ADCSEQ2 ((uint32_t)0x00000400)`
- `#define DMA_Channel_ADCSEQ3 ((uint32_t)0x00000800)`
- `#define DMA_Channel_ADCSEQ4 ((uint32_t)0x00001000)`
- `#define DMA_Channel_ADCSEQ5 ((uint32_t)0x00002000)`
- `#define DMA_Channel_ADCSEQ6 ((uint32_t)0x00004000)`
- `#define DMA_Channel_ADCSEQ7 ((uint32_t)0x00008000)`
- `#define DMA_Channel_SPI0_TX ((uint32_t)0x00010000)`
- `#define DMA_Channel_SPI1_TX ((uint32_t)0x00020000)`
- `#define DMA_Channel_SPI2_TX ((uint32_t)0x00040000)`
- `#define DMA_Channel_SPI3_TX ((uint32_t)0x00080000)`
- `#define DMA_Channel_SPI0_RX ((uint32_t)0x00100000)`
- `#define DMA_Channel_SPI1_RX ((uint32_t)0x00200000)`
- `#define DMA_Channel_SPI2_RX ((uint32_t)0x00400000)`
- `#define DMA_Channel_SPI3_RX ((uint32_t)0x00800000)`

6.33.1 Подробное описание

6.33.2 Макросы

6.33.2.1 `#define DMA_Channel_ADCSEQ0 ((uint32_t)0x00000100)`

Канал DMA секвенсора 0 АЦП

См. определение в файле `niietcm4_dma.h` строка 101

6.33.2.2 `#define DMA_Channel_ADCSEQ1 ((uint32_t)0x00000200)`

Канал DMA секвенсора 1 АЦП

См. определение в файле `niietcm4_dma.h` строка 102

6.33.2.3 `#define DMA_Channel_ADCSEQ2 ((uint32_t)0x00000400)`

Канал DMA секвенсора 2 АЦП

См. определение в файле `niietcm4_dma.h` строка 103

6.33.2.4 `#define DMA_Channel_ADCSEQ3 ((uint32_t)0x00000800)`

Канал DMA секвенсора 3 АЦП

См. определение в файле `niietcm4_dma.h` строка 104

6.33.2.5 `#define DMA_Channel_ADCSEQ4 ((uint32_t)0x00001000)`

Канал DMA секвенсора 4 АЦП

См. определение в файле `niietcm4_dma.h` строка 105

6.33.2.6 `#define DMA_Channel_ADCSEQ5 ((uint32_t)0x00002000)`

Канал DMA секвенсора 5 АЦП

См. определение в файле `niietcm4_dma.h` строка 106

6.33.2.7 `#define DMA_Channel_ADCSEQ6 ((uint32_t)0x00004000)`

Канал DMA секвенсора 6 АЦП

См. определение в файле `niietcm4_dma.h` строка 107

6.33.2.8 `#define DMA_Channel_ADCSEQ7 ((uint32_t)0x00008000)`

Канал DMA секвенсора 7 АЦП

См. определение в файле `niietcm4_dma.h` строка 108

6.33.2.9 `#define DMA_Channel_SPI0_RX ((uint32_t)0x00100000)`

Канал DMA по приему от SPI0

См. определение в файле `niietcm4_dma.h` строка 113

6.33.2.10 `#define DMA_Channel_SPI0_TX ((uint32_t)0x00010000)`

Канал DMA по передаче от SPI0

См. определение в файле `niietcm4_dma.h` строка 109

6.33.2.11 `#define DMA_Channel_SPI1_RX ((uint32_t)0x00200000)`

Канал DMA по приему от SPI1

См. определение в файле `niietcm4_dma.h` строка 114

6.33.2.12 `#define DMA_Channel_SPI1_TX ((uint32_t)0x00020000)`

Канал DMA по передаче от SPI1

См. определение в файле `niietcm4_dma.h` строка 110

6.33.2.13 `#define DMA_Channel_SPI2_RX ((uint32_t)0x00400000)`

Канал DMA по приему от SPI2

См. определение в файле `niietcm4_dma.h` строка 115

6.33.2.14 `#define DMA_Channel_SPI2_TX ((uint32_t)0x00040000)`

Канал DMA по передаче от SPI2

См. определение в файле `niietcm4_dma.h` строка 111

6.33.2.15 `#define DMA_Channel_SPI3_RX ((uint32_t)0x00800000)`

Канал DMA по приему от SPI3

См. определение в файле `niietcm4_dma.h` строка 116

6.33.2.16 `#define DMA_Channel_SPI3_TX ((uint32_t)0x00800000)`

Канал DMA по передаче от SPI3

См. определение в файле `niietcm4_dma.h` строка 112

6.33.2.17 `#define DMA_Channel_UART0_RX ((uint32_t)0x00000010)`

Канал DMA по приему от UART0

См. определение в файле `niietcm4_dma.h` строка 97

6.33.2.18 `#define DMA_Channel_UART0_TX ((uint32_t)0x00000001)`

Канал DMA по передаче от UART0

См. определение в файле `niietcm4_dma.h` строка 93

6.33.2.19 `#define DMA_Channel_UART1_RX ((uint32_t)0x00000020)`

Канал DMA по приему от UART1

См. определение в файле `niietcm4_dma.h` строка 98

6.33.2.20 `#define DMA_Channel_UART1_TX ((uint32_t)0x00000002)`

Канал DMA по передаче от UART1

См. определение в файле `niietcm4_dma.h` строка 94

6.33.2.21 `#define DMA_Channel_UART2_RX ((uint32_t)0x00000040)`

Канал DMA по приему от UART2

См. определение в файле `niietcm4_dma.h` строка 99

6.33.2.22 `#define DMA_Channel_UART2_TX ((uint32_t)0x00000004)`

Канал DMA по передаче от UART2

См. определение в файле `niietcm4_dma.h` строка 95

6.33.2.23 `#define DMA_Channel_UART3_RX ((uint32_t)0x00000080)`

Канал DMA по приему от UART3

См. определение в файле `niietcm4_dma.h` строка 100

6.33.2.24 `#define DMA_Channel_UART3_TX ((uint32_t)0x00000008)`

Канал DMA по передаче от UART3

См. определение в файле `niietcm4_dma.h` строка 96

6.34 Маски каналов по номеру

Макросы

- `#define DMA_Channel_0 ((uint32_t)0x00000001)`
- `#define DMA_Channel_1 ((uint32_t)0x00000002)`
- `#define DMA_Channel_2 ((uint32_t)0x00000004)`
- `#define DMA_Channel_3 ((uint32_t)0x00000008)`
- `#define DMA_Channel_4 ((uint32_t)0x00000010)`
- `#define DMA_Channel_5 ((uint32_t)0x00000020)`
- `#define DMA_Channel_6 ((uint32_t)0x00000040)`
- `#define DMA_Channel_7 ((uint32_t)0x00000080)`
- `#define DMA_Channel_8 ((uint32_t)0x00000100)`
- `#define DMA_Channel_9 ((uint32_t)0x00000200)`
- `#define DMA_Channel_10 ((uint32_t)0x00000400)`
- `#define DMA_Channel_11 ((uint32_t)0x00000800)`
- `#define DMA_Channel_12 ((uint32_t)0x00001000)`
- `#define DMA_Channel_13 ((uint32_t)0x00002000)`
- `#define DMA_Channel_14 ((uint32_t)0x00004000)`
- `#define DMA_Channel_15 ((uint32_t)0x00008000)`
- `#define DMA_Channel_16 ((uint32_t)0x00010000)`
- `#define DMA_Channel_17 ((uint32_t)0x00020000)`
- `#define DMA_Channel_18 ((uint32_t)0x00040000)`
- `#define DMA_Channel_19 ((uint32_t)0x00080000)`
- `#define DMA_Channel_20 ((uint32_t)0x00100000)`
- `#define DMA_Channel_21 ((uint32_t)0x00200000)`
- `#define DMA_Channel_22 ((uint32_t)0x00400000)`
- `#define DMA_Channel_23 ((uint32_t)0x00800000)`

6.34.1 Подробное описание

6.34.2 Макросы

6.34.2.1 `#define DMA_Channel_0 ((uint32_t)0x00000001)`

Канал DMA 0

См. определение в файле `niietcm4_dma.h` строка 126

6.34.2.2 `#define DMA_Channel_1 ((uint32_t)0x00000002)`

Канал DMA 1

См. определение в файле `niietcm4_dma.h` строка 127

6.34.2.3 `#define DMA_Channel_10 ((uint32_t)0x00000400)`

Канал DMA 10

См. определение в файле `niietcm4_dma.h` строка 136

6.34.2.4 `#define DMA_Channel_11 ((uint32_t)0x00000800)`

Канал DMA 11

См. определение в файле `niietcm4_dma.h` строка 137

6.34.2.5 `#define DMA_Channel_12 ((uint32_t)0x00001000)`

Канал DMA 12

См. определение в файле `niietcm4_dma.h` строка 138

6.34.2.6 `#define DMA_Channel_13 ((uint32_t)0x00002000)`

Канал DMA 13

См. определение в файле `niietcm4_dma.h` строка 139

6.34.2.7 `#define DMA_Channel_14 ((uint32_t)0x00004000)`

Канал DMA 14

См. определение в файле `niietcm4_dma.h` строка 140

6.34.2.8 `#define DMA_Channel_15 ((uint32_t)0x00008000)`

Канал DMA 15

См. определение в файле `niietcm4_dma.h` строка 141

6.34.2.9 `#define DMA_Channel_16 ((uint32_t)0x00010000)`

Канал DMA 16

См. определение в файле `niietcm4_dma.h` строка 142

6.34.2.10 `#define DMA_Channel_17 ((uint32_t)0x00020000)`

Канал DMA 17

См. определение в файле `niietcm4_dma.h` строка 143

6.34.2.11 `#define DMA_Channel_18 ((uint32_t)0x00040000)`

Канал DMA 18

См. определение в файле `niietcm4_dma.h` строка 144

6.34.2.12 `#define DMA_Channel_19 ((uint32_t)0x00080000)`

Канал DMA 19

См. определение в файле `niietcm4_dma.h` строка 145

6.34.2.13 `#define DMA_Channel_2 ((uint32_t)0x00000004)`

Канал DMA 2

См. определение в файле `niietcm4_dma.h` строка 128

6.34.2.14 `#define DMA_Channel_20 ((uint32_t)0x00100000)`

Канал DMA 20

См. определение в файле `niietcm4_dma.h` строка 146

6.34.2.15 `#define DMA_Channel_21 ((uint32_t)0x00200000)`

Канал DMA 21

См. определение в файле `niietcm4_dma.h` строка 147

6.34.2.16 `#define DMA_Channel_22 ((uint32_t)0x00400000)`

Канал DMA 22

См. определение в файле `niietcm4_dma.h` строка 148

6.34.2.17 `#define DMA_Channel_23 ((uint32_t)0x00800000)`

Канал DMA 23

См. определение в файле `niietcm4_dma.h` строка 149

6.34.2.18 `#define DMA_Channel_3 ((uint32_t)0x00000008)`

Канал DMA 3

См. определение в файле `niietcm4_dma.h` строка 129

6.34.2.19 `#define DMA_Channel_4 ((uint32_t)0x00000010)`

Канал DMA 4

См. определение в файле `niietcm4_dma.h` строка 130

6.34.2.20 `#define DMA_Channel_5 ((uint32_t)0x00000020)`

Канал DMA 5

См. определение в файле `niietcm4_dma.h` строка 131

6.34.2.21 `#define DMA_Channel_6 ((uint32_t)0x00000040)`

Канал DMA 6

См. определение в файле `niietcm4_dma.h` строка 132

6.34.2.22 `#define DMA_Channel_7 ((uint32_t)0x00000080)`

Канал DMA 7

См. определение в файле `niietcm4_dma.h` строка 133

6.34.2.23 `#define DMA_Channel_8 ((uint32_t)0x00000100)`

Канал DMA 8

См. определение в файле `niietcm4_dma.h` строка 134

6.34.2.24 `#define DMA_Channel_9 ((uint32_t)0x00000200)`

Канал DMA 9

См. определение в файле `niietcm4_dma.h` строка 135

6.35 Типы

Структуры данных

- struct [_CHANNEL_CFG_bits](#)
Битовый доступ к регистру CHANNEL_CFG в [DMA_Channel_TypeDef](#).
- struct [DMA_Channel_TypeDef](#)
Тип, описывающий структуру канала DMA.
- struct [DMA_ConfigStruct_TypeDef](#)
Управляющая структура данных DMA.
- struct [DMA_ConfigData_TypeDef](#)
Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ.
- struct [DMA_Protect_TypeDef](#)
Защита шины при чтении из источника или записи в приемник через DMA.
- struct [DMA_ChannelInit_TypeDef](#)
Структура инициализации канала DMA.
- struct [DMA_Init_TypeDef](#)
Структура инициализации контроллера DMA.

Макросы

- [#define IS_DMA_MODE\(MODE\)](#)
Макрос проверки аргументов типа [DMA_Mode_TypeDef](#).
- [#define IS_DMA_ARBITRATION_RATE\(ARBITRATION_RATE\)](#)
Макрос проверки аргументов типа [DMA_ArbitrationRate_TypeDef](#).
- [#define IS_DMA_DATA_SIZE\(DATA_SIZE\)](#)
Макрос проверки аргументов типа [DMA_DataSize_TypeDef](#).
- [#define IS_DMA_DATA_INC\(DATA_INC\)](#)
Макрос проверки аргументов типа [DMA_DataSize_TypeDef](#).
- [#define IS_DMA_TRANSFERS_TOTAL\(TRANSFERS_TOTAL\) \(\(\(TRANSFERS_TOTAL\) <= \(\(uint32_t\)1024\)\) && \(\(TRANSFERS_TOTAL\) >= \(\(uint32_t\)1\)\)\)](#)
Макрос проверки соответствия величины DMA_TransfersTotal из [DMA_ChannelInit_TypeDef](#) разрешенному диапазону.
- [#define IS_DMA_STATE\(STATE\)](#)
Макрос проверки аргументов типа [DMA_State_TypeDef](#).

Перечисления

- enum [DMA_Mode_TypeDef](#) {
DMA_Mode_Disable, DMA_Mode_Basic, DMA_Mode_AutoReq, DMA_Mode_PingPong,
DMA_Mode_PrmMemScatGath, DMA_Mode_AltMemScatGath, DMA_Mode_PrmPeriph↵
ScatGath, DMA_Mode_AltPeriphScatGath }
Выбор режима работы DMA.
- enum [DMA_ArbitrationRate_TypeDef](#) {
DMA_ArbitrationRate_1, DMA_ArbitrationRate_2, DMA_ArbitrationRate_4, DMA_↵
ArbitrationRate_8,
DMA_ArbitrationRate_16, DMA_ArbitrationRate_32, DMA_ArbitrationRate_64, DMA_↵
ArbitrationRate_128,
DMA_ArbitrationRate_256, DMA_ArbitrationRate_512, DMA_ArbitrationRate_1024 }
Выбор количества передач до выполнения перееарбитрации.
- enum [DMA_DataSize_TypeDef](#) { DMA_DataSize_8, DMA_DataSize_16, DMA_DataSize_32
}

Разрядность данных источника или приемника

- enum DMA_DataInc_TypeDef { DMA_DataInc_8, DMA_DataInc_16, DMA_DataInc_32, DMA_DataInc_Disable }

Шаг инкремента адреса источника при чтении или приемника при записи

- enum DMA_State_TypeDef {
DMA_State_Free, DMA_State_ReadConfigData, DMA_State_ReadSrcDataEndPtr, DMA_State_ReadDstDataEndPtr,
DMA_State_ReadSrcData, DMA_State_WriteDstData, DMA_State_WaitReq, DMA_State_WriteConfigData,
DMA_State_Pause, DMA_State_Done, DMA_State_PeriphScatGath }

Возможные состояния конечного автомата управления контроллером DMA.

6.35.1 Подробное описание

6.35.2 Макросы

6.35.2.1 #define IS_DMA_ARBITRATION_RATE(ARBITRATION_RATE)

Макроопределение:

```
((ARBITRATION_RATE) == DMA_ArbitrationRate_1) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_2) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_4) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_8) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_16) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_32) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_64) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_128) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_256) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_512) || \
((ARBITRATION_RATE) ==
DMA_ArbitrationRate_1024))
```

Макрос проверки аргументов типа DMA_ArbitrationRate_TypeDef.

См. определение в файле niietcm4_dma.h строка 316

Используется в DMA_ChannelInit().

6.35.2.2 #define IS_DMA_DATA_INC(DATA_INC)

Макроопределение:

```
((DATA_INC) == DMA_DataInc_8) || \
((DATA_INC) == DMA_DataInc_16) || \
((DATA_INC) == DMA_DataInc_32) || \
((DATA_INC) == DMA_DataInc_Disable))
```

Макрос проверки аргументов типа DMA_DataSize_TypeDef.

См. определение в файле niietcm4_dma.h строка 374

Используется в DMA_ChannelInit().

6.35.2.3 #define IS_DMA_DATA_SIZE(DATA_SIZE)

Макроопределение:

```
((DATA_SIZE) == DMA_DataSize_8) || \
((DATA_SIZE) == DMA_DataSize_16) || \
((DATA_SIZE) == DMA_DataSize_32))
```

Макрос проверки аргументов типа [DMA_DataSize_TypeDef](#).

См. определение в файле `niietcm4_dma.h` строка 354

Используется в `DMA_ChannelInit()`.

6.35.2.4 #define IS_DMA_MODE(MODE)

Макроопределение:

```
((MODE) == DMA_Mode_Disable) || \
((MODE) == DMA_Mode_Basic) || \
((MODE) == DMA_Mode_AutoReq) || \
((MODE) == DMA_Mode_PingPong) || \
((MODE) == DMA_Mode_PrmMemScatGath) || \
((MODE) == DMA_Mode_AltMemScatGath) || \
((MODE) == DMA_Mode_PrmPeriphScatGath) || \
((MODE) == DMA_Mode_AltPeriphScatGath))
```

Макрос проверки аргументов типа [DMA_Mode_TypeDef](#).

См. определение в файле `niietcm4_dma.h` строка 284

Используется в `DMA_ChannelInit()`.

6.35.2.5 #define IS_DMA_STATE(STATE)

Макроопределение:

```
((STATE) == DMA_State_Free) || \
((STATE) == DMA_State_ReadConfigData) || \
((STATE) == DMA_State_ReadSrcDataEndPtr) || \
((STATE) == DMA_State_ReadDstDataEndPtr) || \
((STATE) == DMA_State_ReadSrcData) || \
((STATE) == DMA_State_WriteDstData) || \
((STATE) == DMA_State_WaitReq) || \
((STATE) == DMA_State_Pause) || \
((STATE) == DMA_State_Done) || \
((STATE) == DMA_State_PeriphScatGath))
```

Макрос проверки аргументов типа [DMA_State_TypeDef](#).

См. определение в файле `niietcm4_dma.h` строка 459

6.35.3 Перечисления

6.35.3.1 enum DMA_ArbitrationRate_TypeDef

Выбор количества передач до выполнения переарбитрации.

Элементы перечислений

<code>DMA_ArbitrationRate_1</code>	Переарбитрация каждую передачу DMA
<code>DMA_ArbitrationRate_2</code>	Переарбитрация каждые 2 передачи DMA
<code>DMA_ArbitrationRate_4</code>	Переарбитрация каждые 4 передачи DMA
<code>DMA_ArbitrationRate_8</code>	Переарбитрация каждые 8 передач DMA

DMA_ArbitrationRate_16 Переарбитрация каждые 16 передач DMA
 DMA_ArbitrationRate_32 Переарбитрация каждые 32 передачи DMA
 DMA_ArbitrationRate_64 Переарбитрация каждые 64 передачи DMA
 DMA_ArbitrationRate_128 Переарбитрация каждые 128 передач DMA
 DMA_ArbitrationRate_256 Переарбитрация каждые 256 передач DMA
 DMA_ArbitrationRate_512 Переарбитрация каждые 512 передач DMA
 DMA_ArbitrationRate_1024 Переарбитрация каждые 1024 передачи DMA

См. определение в файле `niietcm4_dma.h` строка 297

6.35.3.2 enum DMA_DataInc_TypeDef

Шаг инкремента адреса источника при чтении или приемника при записи

Элементы перечислений

DMA_DataInc_8 Инкремент данных 8 бит
 DMA_DataInc_16 Инкремент данных 16 бит
 DMA_DataInc_32 Инкремент данных 32 бит
 DMA_DataInc_Disable Инкремент отсутствует

См. определение в файле `niietcm4_dma.h` строка 362

6.35.3.3 enum DMA_DataSize_TypeDef

Разрядность данных источника или приемника

Элементы перечислений

DMA_DataSize_8 Разрядность данных 8 бит
 DMA_DataSize_16 Разрядность данных 16 бит
 DMA_DataSize_32 Разрядность данных 32 бит

См. определение в файле `niietcm4_dma.h` строка 343

6.35.3.4 enum DMA_Mode_TypeDef

Выбор режима работы DMA.

Элементы перечислений

DMA_Mode_Disable Неактивное состояние
 DMA_Mode_Basic Основной режим передачи
 DMA_Mode_AutoReq Режим передачи с авто-запросом
 DMA_Mode_PingPong Режим передачи "пинг-понг"
 DMA_Mode_PrmMemScatGath Работа с памятью в режиме "разборка-сборка" с использованием первичной управляющей структуры
 DMA_Mode_AltMemScatGath Работа с памятью в режиме "разборка-сборка" с использованием альтернативной управляющей структуры
 DMA_Mode_PrmPeriphScatGath Работа с периферией в режиме "разборка-сборка" с использованием первичной управляющей структуры
 DMA_Mode_AltPeriphScatGath Работа с периферией в режиме "разборка-сборка" с использованием альтернативной управляющей структуры

См. определение в файле `niietcm4_dma.h` строка 268

6.35.3.5 enum DMA_State_TypeDef

Возможные состояния конечного автомата управления контроллером DMA.

Элементы перечислений

DMA_State_Free В покое.

DMA_State_ReadConfigData Чтение управляющих данных канала.

DMA_State_ReadSrcDataEndPtr Чтение указателя конца данных источника.

DMA_State_ReadDstDataEndPtr Чтение указателя конца данных приемника.

DMA_State_ReadSrcData Чтение данных источника.

DMA_State_WriteDstData Запись данных в приемник.

DMA_State_WaitReq Ожидание запроса на выполнение прямого доступа.

DMA_State_WriteConfigData Запись управляющих данных канала.

DMA_State_Pause Приостановлен.

DMA_State_Done Выполнен.

DMA_State_PeriphScatGath Работа с периферией в режиме "разборка-сборка".

См. определение в файле niietcm4_dma.h строка 440

6.36 Функции

Группы

- [Инициализация каналов DMA](#)
- [Инициализация контроллера DMA](#)
- [Конфигурация контроллера DMA](#)
- [Статусная информация](#)

6.36.1 Подробное описание

6.37 Инициализация каналов DMA

Функции

- void [DMA_ChannelDeInit](#) ([DMA_Channel_TypeDef](#) *DMA_Channel)
Деинициализация канала DMA.
- void [DMA_ChannelInit](#) ([DMA_Channel_TypeDef](#) *DMA_Channel, [DMA_ChannelInit_TypeDef](#) *DMA_ChannelInitStruct)
Инициализация канала DMA.
- void [DMA_ChannelStructInit](#) ([DMA_ChannelInit_TypeDef](#) *DMA_ChannelInitStruct)
Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

6.37.1 Подробное описание

6.37.2 Функции

6.37.2.1 void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)

Деинициализация канала DMA.

Аргументы

DMA_Channel	Указатель на структуру типа DMA_Channel_TypeDef , которая содержит конфигурационную информацию канала.
-------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 87

Перекрестные ссылки [DMA_Channel_TypeDef::CHANNEL_CFG](#), [DMA_Channel_TypeDef::DST_DATA_END](#) и [DMA_Channel_TypeDef::SRC_DATA_END](#).

6.37.2.2 void DMA_ChannelInit (DMA_Channel_TypeDef * DMA_Channel, DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Инициализация канала DMA.

Аргументы

DMA_Channel	Непосредственно сама структура канала.
DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которая содержит конфигурационную информацию канала.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 102

Перекрестные ссылки [DMA_Protect_TypeDef::BUFFERABLE](#), [DMA_Protect_TypeDef::CACHEABLE](#), [DMA_Channel_TypeDef::CHANNEL_CFG_bit](#), [_CHANNEL_CFG_bits::CYCLE_CTRL](#), [DMA_ChannelInit_TypeDef::DMA_ArbitrationRate](#), [DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr](#), [DMA_ChannelInit_TypeDef::DMA_DstDataInc](#), [DMA_ChannelInit_TypeDef::DMA_DstDataSize](#), [DMA_ChannelInit_TypeDef::DMA_DstProtect](#), [DMA_ChannelInit_TypeDef::DMA_Mode](#), [DMA_ChannelInit_TypeDef::DMA_NextUseburst](#), [DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr](#), [DMA_ChannelInit_TypeDef::DMA_SrcDataInc](#), [DMA_ChannelInit_TypeDef::DMA_SrcDataSize](#), [DMA_ChannelInit_TypeDef::DMA_SrcProtect](#), [DMA_Channel](#)

Init_TypeDef::DMA_TransfersTotal, DMA_Channel_TypeDef::DST_DATA_END, _CHANNEL_CFG_bits::DST_INC, _CHANNEL_CFG_bits::DST_PROT_BUFFERABLE, _CHANNEL_CFG_bits::DST_PROT_CACHEABLE, _CHANNEL_CFG_bits::DST_PROT_PRIVILEGED, _CHANNEL_CFG_bits::DST_SIZE, IS_DMA_ARBITRATION_RATE, IS_DMA_DATA_INC, IS_DMA_DATA_SIZE, IS_DMA_MODE, IS_DMA_TRANSFERS_TOTAL, IS_FUNCTIONAL_STATE, _CHANNEL_CFG_bits::N_MINUS_1, _CHANNEL_CFG_bits::NEXT_USEBURST, DMA_Protect_TypeDef::PRIVELGED, _CHANNEL_CFG_bits::R_POWER, DMA_Channel_TypeDef::SRC_DATA_END, _CHANNEL_CFG_bits::SRC_INC, _CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE, _CHANNEL_CFG_bits::SRC_PROT_CACHEABLE, _CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED и _CHANNEL_CFG_bits::SRC_SIZE.

6.37.2.3 void DMA_ChannelStructInit (DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

Аргументы

DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которую необходимо проинициализировать.
-----------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 146

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ArbitrationRate_1, DMA_DataInc_Disable, DMA_DataSize_8, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_Mode_Disable, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal и DMA_Protect_TypeDef::PRIVELGED.

6.38 Инициализация контроллера DMA

Функции

- void [DMA_DeInit](#) ()
Деинициализация контроллера DMA.
- void [DMA_Init](#) ([DMA_Init_TypeDef](#) *DMA_InitStruct)
Инициализация контроллера DMA.
- void [DMA_StructInit](#) ([DMA_Init_TypeDef](#) *DMA_InitStruct)
Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

6.38.1 Подробное описание

6.38.2 Функции

6.38.2.1 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 174

6.38.2.2 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

Прежде чем инициализировать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать базовый адрес управляющей структуры с помощью [DMA_BasePtrConfig](#).

Аргументы

DMA_InitStruct	Указатель на структуру типа DMA_Init_TypeDef , которая содержит конфигурационную информацию.
----------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 195

Перекрестные ссылки [DMA_Init_TypeDef::DMA_Channel](#), [DMA_Init_TypeDef::DMA_ChannelEnable](#), [DMA_ChannelEnableCmd\(\)](#), [DMA_Init_TypeDef::DMA_HighPriority](#), [DMA_HighPriorityCmd\(\)](#), [DMA_Init_TypeDef::DMA_PrmAlt](#), [DMA_PrmAltCmd\(\)](#), [DMA_Init_TypeDef::DMA_Protection](#), [DMA_ProtectionConfig\(\)](#), [DMA_Init_TypeDef::DMA_ReqMask](#), [DMA_ReqMaskCmd\(\)](#), [DMA_Init_TypeDef::DMA_UseBurst](#) и [DMA_UseBurstCmd\(\)](#).

6.38.2.3 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

Аргументы

DMA_Init↔ Struct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
---------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 212

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHE↔ABLE`, `DMA_Init_TypeDef::DMA_Channel`, `DMA_Init_TypeDef::DMA_ChannelEnable`, `DMA_↔Init_TypeDef::DMA_HighPriority`, `DMA_Init_TypeDef::DMA_PrmAlt`, `DMA_Init_TypeDef::DM↔A_Protection`, `DMA_Init_TypeDef::DMA_ReqMask`, `DMA_Init_TypeDef::DMA_UseBurst` и `DM↔A_Protect_TypeDef::PRIVELGED`.

6.39 Конфигурация контроллера DMA

Функции

- void [DMA_BasePtrConfig](#) (uint32_t BasePtr)
Установка базового адреса управляющих каналов.
- void [DMA_ProtectionConfig](#) (DMA_Protect_TypeDef *DMA_Protection)
Управление защитой шины при обращении DMA к управляющим данным.
- void [DMA_MasterEnableCmd](#) (FunctionalState State)
Разрешения работы контроллера DMA.
- void [DMA_SWRequestCmd](#) (uint32_t DMA_Channel)
Программный запрос на осуществление передач DMA по выбранным каналам.
- void [DMA_UseBurstCmd](#) (uint32_t DMA_Channel, FunctionalState State)
Установка пакетного обмена каналов DMA.
- void [DMA_ReqMaskCmd](#) (uint32_t DMA_Channel, FunctionalState State)
Маскирование каналов DMA.
- void [DMA_ChannelEnableCmd](#) (uint32_t DMA_Channel, FunctionalState State)
Активация каналов DMA.
- void [DMA_PrmAltCmd](#) (uint32_t DMA_Channel, FunctionalState State)
Установка первичной/альтернативной управляющей структуры каналов DMA.
- void [DMA_HighPriorityCmd](#) (uint32_t DMA_Channel, FunctionalState State)
Установка высокого приоритета каналов DMA.

6.39.1 Подробное описание

6.39.2 Функции

6.39.2.1 void DMA_BasePtrConfig (uint32_t BasePtr)

Установка базового адреса управляющих каналов.

Аргументы

BasePtr	Значение базового адреса.
---------	---------------------------

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 231

6.39.2.2 void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)

Активация каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 373

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.39.2.3 `void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка высокого приоритета каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 421

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.39.2.4 `void DMA_MasterEnableCmd (FunctionalState State)`

Разрешения работы контроллера DMA.

Внимание

Прежде чем включать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать контроллер DMA через функцию инициализации [DMA_Init](#) или вручную - [Конфигурация контроллера DMA](#).

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 287

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

6.39.2.5 `void DMA_PrmAltCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
-------------	--

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 397

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.39.2.6 `void DMA_ProtectionConfig (DMA_Protect_TypeDef * DMA_Protection)`

Управление защитой шины при обращении DMA к управляющим данным.

Аргументы

DMA_↔ Protection	Структура, содержащая конфигурацию защиты. Параметр принимает структуру типа DMA_Protect_TypeDef .
---------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 243

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHE↔ABLE`, `IS_FUNCTIONAL_STATE` и `DMA_Protect_TypeDef::PRIVELGED`.

Используется в `DMA_Init()`.

6.39.2.7 `void DMA_ReqMaskCmd (uint32_t DMA_Channel, FunctionalState State)`

Маскирование каналов DMA.

Внимание

По маскированным каналам игнорируются запросы на передачи.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 349

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.39.2.8 `void DMA_SWRequestCmd (uint32_t DMA_Channel)`

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
-------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 308

Перекрестные ссылки IS_DMA_CHANNEL.

6.39.2.9 void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)

Установка пакетного обмена каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 324

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

6.40 Статусная информация

Функции

- [DMA_State_TypeDef DMA_StateStatus \(\)](#)
Доступ к текущему конечного автомата контроллера DMA.
- [FunctionalState DMA_MasterEnableStatus \(\)](#)
Состояние контроллера DMA.
- [FunctionalState DMA_WaitOnReqStatus \(uint32_t DMA_Channel\)](#)
Показывает поддерживает ли канал одиночные SREQ запросы.
- [OperationStatus DMA_ErrorStatus \(\)](#)
Показывает наличие ошибки на шине.
- [void DMA_ClearErrorStatus \(\)](#)
Сброс флага ошибки на шине.

6.40.1 Подробное описание

6.40.2 Функции

6.40.2.1 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 524

6.40.2.2 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

Status	Одно из значений OperationStatus: <ul style="list-style-type: none"> • ОК - ошибок не было; • ERROR - произошла ошибка.
--------	---

См. определение в файле niietcm4_dma.c строка 503

6.40.2.3 FunctionalState DMA_MasterEnableStatus ()

Состояние контроллера DMA.

Возвращаемые значения

Status	Текущее состояние контроллера DMA.
--------	------------------------------------

См. определение в файле niietcm4_dma.c строка 455

6.40.2.4 DMA_State_TypeDef DMA_StateStatus ()

Доступ к текущему конечного автомата контроллера DMA.

Возвращаемые значения

State	Текущее состояние конечного автомата.
-------	---------------------------------------

См. определение в файле `niietcm4_dma.c` строка 441

6.40.2.5 FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

Status	<p>Одно из значений FunctionalState:</p> <ul style="list-style-type: none"> • ENABLE - поддерживаются SREQ (как и блочные BREQ); • DISABLE - поддерживаются только блочные запросы BREQ.
--------	--

См. определение в файле `niietcm4_dma.c` строка 478

Перекрестные ссылки IS_GET_DMA_CHANNEL.

6.41 Константы

Группы

- [Маски адреса](#)

6.41.1 Подробное описание

6.42 Маски адреса

Макросы

- `#define EXTMEM_CEMask_Addr_11 ((uint32_t)0x0001)`
- `#define EXTMEM_CEMask_Addr_12 ((uint32_t)0x0002)`
- `#define EXTMEM_CEMask_Addr_13 ((uint32_t)0x0004)`
- `#define EXTMEM_CEMask_Addr_14 ((uint32_t)0x0008)`
- `#define EXTMEM_CEMask_Addr_15 ((uint32_t)0x0010)`
- `#define EXTMEM_CEMask_Addr_16 ((uint32_t)0x0020)`
- `#define EXTMEM_CEMask_Addr_17 ((uint32_t)0x0040)`
- `#define EXTMEM_CEMask_Addr_18 ((uint32_t)0x0080)`
- `#define EXTMEM_CEMask_Addr_19 ((uint32_t)0x0100)`
- `#define EXTMEM_CEMask_Addr_11_19 ((uint32_t)0x01FF)`
- `#define IS_EXTMEM_CE_MASK(CE_MASK) (((CE_MASK) & ((uint32_t)0xFFFFFE00)) == ((uint32_t)0x00))`

Макрос проверки соответствия маски адреса разрешенному диапазону.

6.42.1 Подробное описание

6.42.2 Макросы

6.42.2.1 `#define EXTMEM_CEMask_Addr_11 ((uint32_t)0x0001)`

Маска бита 11 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 56

6.42.2.2 `#define EXTMEM_CEMask_Addr_11_19 ((uint32_t)0x01FF)`

Маски [19:11] битов адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 65

6.42.2.3 `#define EXTMEM_CEMask_Addr_12 ((uint32_t)0x0002)`

Маска бита 12 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 57

6.42.2.4 `#define EXTMEM_CEMask_Addr_13 ((uint32_t)0x0004)`

Маска бита 13 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 58

6.42.2.5 `#define EXTMEM_CEMask_Addr_14 ((uint32_t)0x0008)`

Маска бита 14 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 59

6.42.2.6 `#define EXTMEM_CEMask_Addr_15 ((uint32_t)0x0010)`

Маска бита 15 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 60

6.42.2.7 `#define EXTMEM_CEMask_Addr_16 ((uint32_t)0x0020)`

Маска бита 16 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 61

6.42.2.8 `#define EXTMEM_CEMask_Addr_17 ((uint32_t)0x0040)`

Маска бита 17 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 62

6.42.2.9 `#define EXTMEM_CEMask_Addr_18 ((uint32_t)0x0080)`

Маска бита 18 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 63

6.42.2.10 `#define EXTMEM_CEMask_Addr_19 ((uint32_t)0x0100)`

Маска бита 19 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 64

6.43 Типы

Структуры данных

- struct `EXTMEM_Init_TypeDef`
Структура инициализации внешней памяти.

Макросы

- #define `IS_EXTMEM_WIDTH(WIDTH)`
Макрос проверки аргументов типа `EXTMEM_Width_TypeDef`.
- #define `IS_EXTMEM_RW_WAITSTATE(WAITSTATE)`
Макрос проверки аргументов типа `EXTMEM_RWWaitState_TypeDef`.
- #define `IS_EXTMEM_WRITE_WAITSTATE(WAITSTATE)`
Макрос проверки аргументов типа `EXTMEM_WriteWaitState_TypeDef`.
- #define `IS_EXTMEM_READ_WAITSTATE(WAITSTATE)`
Макрос проверки аргументов типа `EXTMEM_ReadWaitState_TypeDef`.

Перечисления

- enum `EXTMEM_Width_TypeDef` { `EXTMEM_Width_8bit`, `EXTMEM_Width_16bit` }
Разрядность контроллера внешней памяти.
- enum `EXTMEM_RWWaitState_TypeDef` {
`EXTMEM_RWWaitState_1`, `EXTMEM_RWWaitState_2`, `EXTMEM_RWWaitState_3`, `EXTMEM_RWWaitState_4`,
`EXTMEM_RWWaitState_5`, `EXTMEM_RWWaitState_6`, `EXTMEM_RWWaitState_7`, `EXTMEM_RWWaitState_8` }
Длительность цикла переключения шины в системных тактах.
- enum `EXTMEM_WriteWaitState_TypeDef` {
`EXTMEM_WriteWaitState_1`, `EXTMEM_WriteWaitState_2`, `EXTMEM_WriteWaitState_3`,
`EXTMEM_WriteWaitState_4`,
`EXTMEM_WriteWaitState_5`, `EXTMEM_WriteWaitState_6`, `EXTMEM_WriteWaitState_7`,
`EXTMEM_WriteWaitState_8` }
Длительность цикла записи слова данных в системных тактах.
- enum `EXTMEM_ReadWaitState_TypeDef` {
`EXTMEM_ReadWaitState_1`, `EXTMEM_ReadWaitState_2`, `EXTMEM_ReadWaitState_3`,
`EXTMEM_ReadWaitState_4`,
`EXTMEM_ReadWaitState_5`, `EXTMEM_ReadWaitState_6`, `EXTMEM_ReadWaitState_7`,
`EXTMEM_ReadWaitState_8` }
Длительность цикла чтения слова данных в системных тактах.

6.43.1 Подробное описание

6.43.2 Макросы

6.43.2.1 #define IS_EXTMEM_READ_WAITSTATE(WAITSTATE)

Макроопределение:

```
((WAITSTATE) == EXTMEM_ReadWaitState_1) || \
    ((WAITSTATE) == EXTMEM_ReadWaitState_2) || \
    ((WAITSTATE) == EXTMEM_ReadWaitState_3) || \
    ((WAITSTATE) ==
```

```

EXTMEM_ReadWaitState_4) || \
    ((WAITSTATE) ==
EXTMEM_ReadWaitState_5) || \
    ((WAITSTATE) ==
EXTMEM_ReadWaitState_6) || \
    ((WAITSTATE) ==
EXTMEM_ReadWaitState_7) || \
    ((WAITSTATE) ==
EXTMEM_ReadWaitState_8))

```

Макрос проверки аргументов типа [EXTMEM_ReadWaitState_TypeDef](#).

См. определение в файле niietcm4_extmem.h строка 180

Используется в EXTMEM_Init().

6.43.2.2 #define IS_EXTMEM_RW_WAITSTATE(WAITSTATE)

Макроопределение:

```

(((WAITSTATE) == EXTMEM_RWWaitState_1) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_2) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_3) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_4) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_5) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_6) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_7) || \
    ((WAITSTATE) == EXTMEM_RWWaitState_8))

```

Макрос проверки аргументов типа [EXTMEM_RWWaitState_TypeDef](#).

См. определение в файле niietcm4_extmem.h строка 122

Используется в EXTMEM_Init().

6.43.2.3 #define IS_EXTMEM_WIDTH(WIDTH)

Макроопределение:

```

(((WIDTH) == EXTMEM_Width_8bit) || \
    ((WIDTH) == EXTMEM_Width_16bit))

```

Макрос проверки аргументов типа [EXTMEM_Width_TypeDef](#).

См. определение в файле niietcm4_extmem.h строка 99

Используется в EXTMEM_Init().

6.43.2.4 #define IS_EXTMEM_WRITE_WAITSTATE(WAITSTATE)

Макроопределение:

```

(((WAITSTATE) == EXTMEM_WriteWaitState_1) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_2) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_3) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_4) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_5) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_6) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_7) || \
    ((WAITSTATE) ==
EXTMEM_WriteWaitState_8))

```

Макрос проверки аргументов типа [EXTMEM_WriteWaitState_TypeDef](#).

См. определение в файле `niietcm4_extmem.h` строка 151

Используется в `EXTMEM_Init()`.

6.43.3 Перечисления

6.43.3.1 enum EXTMEM_ReadWaitState_TypeDef

Длительность цикла чтения слова данных в системных тактах.

Элементы перечислений

EXTMEM_ReadWaitState_1 1 цикл ожидания.
EXTMEM_ReadWaitState_2 2 цикла ожидания.
EXTMEM_ReadWaitState_3 3 цикла ожидания.
EXTMEM_ReadWaitState_4 4 цикла ожидания.
EXTMEM_ReadWaitState_5 5 циклов ожидания.
EXTMEM_ReadWaitState_6 6 циклов ожидания.
EXTMEM_ReadWaitState_7 7 циклов ожидания.
EXTMEM_ReadWaitState_8 8 циклов ожидания.

См. определение в файле `niietcm4_extmem.h` строка 164

6.43.3.2 enum EXTMEM_RWWaitState_TypeDef

Длительность цикла переключения шины в системных тактах.

Элементы перечислений

EXTMEM_RWWaitState_1 1 цикл ожидания.
EXTMEM_RWWaitState_2 2 цикла ожидания.
EXTMEM_RWWaitState_3 3 цикла ожидания.
EXTMEM_RWWaitState_4 4 цикла ожидания.
EXTMEM_RWWaitState_5 5 циклов ожидания.
EXTMEM_RWWaitState_6 6 циклов ожидания.
EXTMEM_RWWaitState_7 7 циклов ожидания.
EXTMEM_RWWaitState_8 8 циклов ожидания.

См. определение в файле `niietcm4_extmem.h` строка 106

6.43.3.3 enum EXTMEM_Width_TypeDef

Разрядность контроллера внешней памяти.

Элементы перечислений

EXTMEM_Width_8bit 8-разрядный режим работы.
EXTMEM_Width_16bit 16-разрядный режим работы.

См. определение в файле `niietcm4_extmem.h` строка 89

6.43.3.4 enum EXTMEM_WriteWaitState_TypeDef

Длительность цикла записи слова данных в системных тактах.

Элементы перечислений

EXTMEM_WriteWaitState_1 1 цикл ожидания.
EXTMEM_WriteWaitState_2 2 цикла ожидания.
EXTMEM_WriteWaitState_3 3 цикла ожидания.
EXTMEM_WriteWaitState_4 4 цикла ожидания.
EXTMEM_WriteWaitState_5 5 циклов ожидания.
EXTMEM_WriteWaitState_6 6 циклов ожидания.
EXTMEM_WriteWaitState_7 7 циклов ожидания.
EXTMEM_WriteWaitState_8 8 циклов ожидания.

См. определение в файле niietcm4_extmem.h строка 135

6.44 Функции

Функции

- void [EXTMEM_DeInit](#) ()
Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.
- void [EXTMEM_Init](#) ([EXTMEM_Init_TypeDef](#) *[EXTMEM_InitStruct](#))
Инициализирует внешнюю память согласно параметрам структуры [EXTMEM_InitStruct](#).
- void [EXTMEM_StructInit](#) ([EXTMEM_Init_TypeDef](#) *[EXTMEM_InitStruct](#))
Заполнение каждого члена структуры [EXTMEM_InitStruct](#) значениями по умолчанию.

6.44.1 Подробное описание

6.44.2 Функции

6.44.2.1 void [EXTMEM_DeInit](#) ()

Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_extmem.c` строка 121

Перекрестные ссылки [EXT_MEM_CFG_Reset_Value](#).

6.44.2.2 void [EXTMEM_Init](#) ([EXTMEM_Init_TypeDef](#) * [EXTMEM_InitStruct](#))

Инициализирует внешнюю память согласно параметрам структуры [EXTMEM_InitStruct](#).

Аргументы

EXTMEM_InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
-----------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_extmem.c` строка 85

Перекрестные ссылки [IS_EXTMEM_CE_MASK](#), [IS_EXTMEM_READ_WAITSTATE](#), [IS_EXTMEM_RW_WAITSTATE](#), [IS_EXTMEM_WIDTH](#) и [IS_EXTMEM_WRITE_WAITSTATE](#).

6.44.2.3 void [EXTMEM_StructInit](#) ([EXTMEM_Init_TypeDef](#) * [EXTMEM_InitStruct](#))

Заполнение каждого члена структуры [EXTMEM_InitStruct](#) значениями по умолчанию.

Аргументы

EXTMEM_InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
-----------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_extmem.c` строка 107

Перекрестные ссылки `EXTMEM_Init_TypeDef::CEMask`, `EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState`, `EXTMEM_ReadWaitState_8`, `EXTMEM_Init_TypeDef::EXTMEM_RWWaitState`, `EXTMEM_RWWaitState_1`, `EXTMEM_Init_TypeDef::EXTMEM_Width`, `EXTMEM_Width_16bit`, `EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState` и `EXTMEM_WriteWaitState_1`.

6.45 Типы

Структуры данных

- struct `GPIO_Init_TypeDef`
Структура инициализации GPIO.

Макросы

- #define `IS_GPIO_QUAL_PERIOD(PERIOD)` (((PERIOD) & ((uint32_t)0xFFFFF00)) == ((uint32_t)0x00))
Макрос проверки соответствия величины периода фильтрации разрешенному диапазону.
- #define `IS_GPIO_BIT_ACTION(ACTION)` (((ACTION) == `Bit_CLEAR`) || ((ACTION) == `Bit_SET`))
Макрос проверки аргументов типа `BitAction`.
- #define `IS_GPIO_DIR(DIR)`
Макрос проверки аргументов типа `GPIO_Dir_TypeDef`.
- #define `IS_GPIO_MODE(MODE)`
Макрос проверки аргументов типа `GPIO_Mode_TypeDef`.
- #define `IS_GPIO_INT_TYPE(INT_TYPE)`
Макрос проверки аргументов типа `GPIO_IntType_TypeDef`.
- #define `IS_GPIO_INT_POL(INT_POL)`
Макрос проверки аргументов типа `GPIO_IntPol_TypeDef`.
- #define `IS_GPIO_OUT(OUT)`
Макрос проверки аргументов типа `GPIO_Out_TypeDef`.
- #define `IS_GPIO_LOAD(LOAD)`
Макрос проверки аргументов типа `GPIO_Load_TypeDef`.
- #define `IS_GPIO_OUT_MODE(OUT_MODE)`
Макрос проверки аргументов типа `GPIO_OutMode_TypeDef`.
- #define `IS_GPIO_PULLUP(PULLUP)`
Макрос проверки аргументов типа `GPIO_PullUp_TypeDef`.
- #define `IS_GPIO_SYNC(SYNC)`
Макрос проверки аргументов типа `GPIO_Sync_TypeDef`.
- #define `IS_GPIO_QUAL(QUAL)`
Макрос проверки аргументов типа `GPIO_Qual_TypeDef`.
- #define `IS_GPIO_QUAL_MODE(QUAL_MODE)`
Макрос проверки аргументов типа `GPIO_QualMode_TypeDef`.
- #define `IS_GPIO_ALT_FUNC(ALT_FUNC)`
Макрос проверки аргументов типа `GPIO_AltFunc_TypeDef`.

Перечисления

- enum `BitAction` { `Bit_CLEAR` = 0, `Bit_SET` }
Тип, определяющий состояния бита.
- enum `GPIO_Dir_TypeDef` { `GPIO_Dir_In`, `GPIO_Dir_Out` }
Выбор направления работы пина.
- enum `GPIO_Mode_TypeDef` { `GPIO_Mode_IO`, `GPIO_Mode_AltFunc` }
Выбор режима работы пина.
- enum `GPIO_IntType_TypeDef` { `GPIO_IntType_Level`, `GPIO_IntType_Edge` }
Выбор события для возникновения прерывания.
- enum `GPIO_IntPol_TypeDef` { `GPIO_IntPol_Neg`, `GPIO_IntPol_Pos` }

- Выбор полярности события для возникновения прерывания.
- enum `GPIO_Out_TypeDef` { `GPIO_Out_Dis`, `GPIO_Out_En` }
- Включение выхода пина.
- enum `GPIO_Load_TypeDef` { `GPIO_Load_8mA`, `GPIO_Load_16mA` }
- Выбор максимальной нагрузочной способности пина.
- enum `GPIO_OutMode_TypeDef` { `GPIO_OutMode_PP`, `GPIO_OutMode_OD` }
- Выбор режима работы выходных каскадов.
- enum `GPIO_PullUp_TypeDef` { `GPIO_PullUp_Dis`, `GPIO_PullUp_En` }
- Включение подтяжки к питанию.
- enum `GPIO_Sync_TypeDef` { `GPIO_Sync_Dis`, `GPIO_Sync_En` }
- Включение режима пересинхронизации входов через 2 триггера-защелки.
- enum `GPIO_Qual_TypeDef` { `GPIO_Qual_Dis`, `GPIO_Qual_En` }
- Включение входного фильтра.
- enum `GPIO_QualMode_TypeDef` { `GPIO_QualMode_3sample`, `GPIO_QualMode_6sample` }
- Выбор режима работы входного фильтра.
- enum `GPIO_AltFunc_TypeDef` { `GPIO_AltFunc_1`, `GPIO_AltFunc_2`, `GPIO_AltFunc_3` }
- Выбор номера альтернативной функции пина.

6.45.1 Подробное описание

6.45.2 Макросы

6.45.2.1 #define IS_GPIO_ALT_FUNC(ALT_FUNC)

Макроопределение:

```
((ALT_FUNC) == GPIO_AltFunc_1) || \
((ALT_FUNC) == GPIO_AltFunc_2) || \
((ALT_FUNC) == GPIO_AltFunc_3))
```

Макрос проверки аргументов типа `GPIO_AltFunc_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 276

Используется в `GPIO_Init()`.

6.45.2.2 #define IS_GPIO_DIR(DIR)

Макроопределение:

```
((DIR) == GPIO_Dir_In) || \
((DIR) == GPIO_Dir_Out))
```

Макрос проверки аргументов типа `GPIO_Dir_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 88

Используется в `GPIO_Init()`.

6.45.2.3 #define IS_GPIO_INT_POL(INT_POL)

Макроопределение:

```
((INT_POL) == GPIO_IntPol_Neg) || \
((INT_POL) == GPIO_IntPol_Pos))
```

Макрос проверки аргументов типа `GPIO_IntPol_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 139

Используется в `GPIO_ITConfig()`.

6.45.2.4 `#define IS_GPIO_INT_TYPE(INT_TYPE)`

Макроопределение:

```
((INT_TYPE) == GPIO_IntType_Level) || \
((INT_TYPE) == GPIO_IntType_Edge))
```

Макрос проверки аргументов типа `GPIO_IntType_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 122

Используется в `GPIO_ITConfig()`.

6.45.2.5 `#define IS_GPIO_LOAD(LOAD)`

Макроопределение:

```
((LOAD) == GPIO_Load_8mA) || \
((LOAD) == GPIO_Load_16mA))
```

Макрос проверки аргументов типа `GPIO_Load_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 173

Используется в `GPIO_Init()`.

6.45.2.6 `#define IS_GPIO_MODE(MODE)`

Макроопределение:

```
((MODE) == GPIO_Mode_IO) || \
((MODE) == GPIO_Mode_AltFunc))
```

Макрос проверки аргументов типа `GPIO_Mode_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 105

Используется в `GPIO_Init()`.

6.45.2.7 `#define IS_GPIO_OUT(OUT)`

Макроопределение:

```
((OUT) == GPIO_Out_Dis) || \
((OUT) == GPIO_Out_En))
```

Макрос проверки аргументов типа `GPIO_Out_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 156

Используется в `GPIO_Init()`.

6.45.2.8 `#define IS_GPIO_OUT_MODE(OUT_MODE)`

Макроопределение:

```
((OUT_MODE) == GPIO_OutMode_PP) || \
((OUT_MODE) == GPIO_OutMode_OD))
```

Макрос проверки аргументов типа `GPIO_OutMode_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 190

Используется в `GPIO_Init()`.

6.45.2.9 `#define IS_GPIO_PULLUP(PULLUP)`

Макроопределение:

```
((PULLUP) == GPIO_PullUp_Dis) || \
((PULLUP) == GPIO_PullUp_En))
```

Макрос проверки аргументов типа `GPIO_PullUp_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 207

Используется в `GPIO_Init()`.

6.45.2.10 `#define IS_GPIO_QUAL(QUAL)`

Макроопределение:

```
((QUAL) == GPIO_Qual_Dis) || \
((QUAL) == GPIO_Qual_En))
```

Макрос проверки аргументов типа `GPIO_Qual_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 241

6.45.2.11 `#define IS_GPIO_QUAL_MODE(QUAL_MODE)`

Макроопределение:

```
((QUAL_MODE) == GPIO_QualMode_3sample) || \
((QUAL_MODE) == GPIO_QualMode_6sample))
```

Макрос проверки аргументов типа `GPIO_QualMode_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 258

Используется в `GPIO_QualConfig()`.

6.45.2.12 `#define IS_GPIO_SYNC(SYNC)`

Макроопределение:

```
((SYNC) == GPIO_Sync_Dis) || \
((SYNC) == GPIO_Sync_En))
```

Макрос проверки аргументов типа `GPIO_Sync_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 224

6.45.3 Перечисления

6.45.3.1 `enum BitAction`

Тип, определяющий состояния бита.

Элементы перечислений

`Bit_CLEAR` Бит очищен.

`Bit_SET` Бит установлен.

См. определение в файле `niietcm4_gpio.h` строка 62

6.45.3.2 enum GPIO_AltFunc_TypeDef

Выбор номера альтернативной функции пина.

Элементы перечислений

- GPIO_AltFunc_1 Альтернативная функция 1.
- GPIO_AltFunc_2 Альтернативная функция 2.
- GPIO_AltFunc_3 Альтернативная функция 3.

См. определение в файле `niietcm4_gpio.h` строка 265

6.45.3.3 enum GPIO_Dir_TypeDef

Выбор направления работы пина.

Элементы перечислений

- GPIO_Dir_In Пин настроен на вход.
- GPIO_Dir_Out Пин настроен на выход.

См. определение в файле `niietcm4_gpio.h` строка 78

6.45.3.4 enum GPIO_IntPol_TypeDef

Выбор полярности события для возникновения прерывания.

Элементы перечислений

- GPIO_IntPol_Neg Прерывание по низкому уровню или отрицательному фронту.
- GPIO_IntPol_Pos Прерывание по высокому уровню или положительному фронту.

См. определение в файле `niietcm4_gpio.h` строка 129

6.45.3.5 enum GPIO_IntType_TypeDef

Выбор события для возникновения прерывания.

Элементы перечислений

- GPIO_IntType_Level Прерывание по уровню.
- GPIO_IntType_Edge Прерывание по перепаду.

См. определение в файле `niietcm4_gpio.h` строка 112

6.45.3.6 enum GPIO_Load_TypeDef

Выбор максимальной нагрузочной способности пина.

Элементы перечислений

- GPIO_Load_8mA Максимальный ток 8мА.
- GPIO_Load_16mA Максимальный ток 16мА.

См. определение в файле `niietcm4_gpio.h` строка 163

6.45.3.7 enum GPIO_Mode_TypeDef

Выбор режима работы пина.

Элементы перечислений

GPIO_Mode_IO Пин в режиме ввода-вывода.

GPIO_Mode_AltFunc Пин в режиме альтернативной функции.

См. определение в файле niietcm4_gpio.h строка 95

6.45.3.8 enum GPIO_Out_TypeDef

Включение выхода пина.

Элементы перечислений

GPIO_Out_Dis Пин в третьем состоянии.

GPIO_Out_En Пин работает как выход.

См. определение в файле niietcm4_gpio.h строка 146

6.45.3.9 enum GPIO_OutMode_TypeDef

Выбор режима работы выходных каскадов.

Элементы перечислений

GPIO_OutMode_PP Режим пуш-пулл.

GPIO_OutMode_OD Режим открытого коллектора.

См. определение в файле niietcm4_gpio.h строка 180

6.45.3.10 enum GPIO_PullUp_TypeDef

Включение подтяжки к питанию.

Элементы перечислений

GPIO_PullUp_Dis Внутренняя подтяжка к питанию выключена.

GPIO_PullUp_En Внутренняя подтяжка к питанию включена.

См. определение в файле niietcm4_gpio.h строка 197

6.45.3.11 enum GPIO_Qual_TypeDef

Включение входного фильтра.

Элементы перечислений

GPIO_Qual_Dis Входной фильтр выключен.

GPIO_Qual_En Входной фильтр включен.

См. определение в файле niietcm4_gpio.h строка 231

6.45.3.12 enum GPIO_QualMode_TypeDef

Выбор режима работы входного фильтра.

Элементы перечислений

GPIO_QualMode_3sample Используется 3 отсчета для фильтрации.

GPIO_QualMode_6sample Используется 6 отсчетов для фильтрации.

См. определение в файле niietcm4_gpio.h строка 248

6.45.3.13 enum GPIO_Sync_TypeDef

Включение режима пересинхронизации входов через 2 триггера-защелки.

Элементы перечислений

GPIO_Sync_Dis Пересинхронизация входа выключена.

GPIO_Sync_En Пересинхронизация входа включена.

См. определение в файле niietcm4_gpio.h строка 214

6.46 Константы

Группы

- [Маски пинов](#)

6.46.1 Подробное описание

6.47 Маски пинов

Макросы

```

• #define GPIO_Pin_0 ((uint32_t)0x0001)
• #define GPIO_Pin_1 ((uint32_t)0x0002)
• #define GPIO_Pin_2 ((uint32_t)0x0004)
• #define GPIO_Pin_3 ((uint32_t)0x0008)
• #define GPIO_Pin_4 ((uint32_t)0x0010)
• #define GPIO_Pin_5 ((uint32_t)0x0020)
• #define GPIO_Pin_6 ((uint32_t)0x0040)
• #define GPIO_Pin_7 ((uint32_t)0x0080)
• #define GPIO_Pin_8 ((uint32_t)0x0100)
• #define GPIO_Pin_9 ((uint32_t)0x0200)
• #define GPIO_Pin_10 ((uint32_t)0x0400)
• #define GPIO_Pin_11 ((uint32_t)0x0800)
• #define GPIO_Pin_12 ((uint32_t)0x1000)
• #define GPIO_Pin_13 ((uint32_t)0x2000)
• #define GPIO_Pin_14 ((uint32_t)0x4000)
• #define GPIO_Pin_15 ((uint32_t)0x8000)
• #define GPIO_Pin_0_3 ((uint32_t)0x000F)
• #define GPIO_Pin_4_7 ((uint32_t)0x00F0)
• #define GPIO_Pin_8_11 ((uint32_t)0x0F00)
• #define GPIO_Pin_12_15 ((uint32_t)0xF000)
• #define GPIO_Pin_0_7 ((uint32_t)0x00FF)
• #define GPIO_Pin_8_15 ((uint32_t)0xFF00)
• #define GPIO_Pin_All ((uint32_t)0xFFFF)
• #define IS_GPIO_PIN(PIN) (((PIN) != (uint32_t)0x0000) && (((PIN) & (uint32_t)0xFFFF) <= 0xFFFF))

```

Макрос проверки номеров пинов на попадание в допустимый диапазон.

```
• #define IS_GET_GPIO_PIN(PIN)
```

Макрос проверки номера пина при работе с пинами по отдельности.

6.47.1 Подробное описание

6.47.2 Макросы

6.47.2.1 #define GPIO_Pin_0 ((uint32_t)0x0001)

Пин 0 выбран.

См. определение в файле n1ietcm4_gpio.h строка 319

Используется в RCC_SysClkDiv2Out().

6.47.2.2 #define GPIO_Pin_0_3 ((uint32_t)0x000F)

Пины 0-3 выбраны.

См. определение в файле n1ietcm4_gpio.h строка 335

6.47.2.3 #define GPIO_Pin_0_7 ((uint32_t)0x00FF)

Пины 0-7 выбраны.

См. определение в файле n1ietcm4_gpio.h строка 339

6.47.2.4 `#define GPIO_Pin_1 ((uint32_t)0x0002)`

Пин 1 выбран.

См. определение в файле `niietcm4_gpio.h` строка 320

6.47.2.5 `#define GPIO_Pin_10 ((uint32_t)0x0400)`

Пин 10 выбран.

См. определение в файле `niietcm4_gpio.h` строка 329

6.47.2.6 `#define GPIO_Pin_11 ((uint32_t)0x0800)`

Пин 11 выбран.

См. определение в файле `niietcm4_gpio.h` строка 330

6.47.2.7 `#define GPIO_Pin_12 ((uint32_t)0x1000)`

Пин 12 выбран.

См. определение в файле `niietcm4_gpio.h` строка 331

6.47.2.8 `#define GPIO_Pin_12_15 ((uint32_t)0xF000)`

Пины 12-15 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 338

6.47.2.9 `#define GPIO_Pin_13 ((uint32_t)0x2000)`

Пин 13 выбран.

См. определение в файле `niietcm4_gpio.h` строка 332

6.47.2.10 `#define GPIO_Pin_14 ((uint32_t)0x4000)`

Пин 14 выбран.

См. определение в файле `niietcm4_gpio.h` строка 333

6.47.2.11 `#define GPIO_Pin_15 ((uint32_t)0x8000)`

Пин 15 выбран.

См. определение в файле `niietcm4_gpio.h` строка 334

6.47.2.12 `#define GPIO_Pin_2 ((uint32_t)0x0004)`

Пин 2 выбран.

См. определение в файле `niietcm4_gpio.h` строка 321

6.47.2.13 `#define GPIO_Pin_3 ((uint32_t)0x0008)`

Пин 3 выбран.

См. определение в файле niietcm4_gpio.h строка 322

6.47.2.14 #define GPIO_Pin_4 ((uint32_t)0x0010)

Пин 4 выбран.

См. определение в файле niietcm4_gpio.h строка 323

6.47.2.15 #define GPIO_Pin_4_7 ((uint32_t)0x00F0)

Пины 4-7 выбраны.

См. определение в файле niietcm4_gpio.h строка 336

6.47.2.16 #define GPIO_Pin_5 ((uint32_t)0x0020)

Пин 5 выбран.

См. определение в файле niietcm4_gpio.h строка 324

6.47.2.17 #define GPIO_Pin_6 ((uint32_t)0x0040)

Пин 6 выбран.

См. определение в файле niietcm4_gpio.h строка 325

6.47.2.18 #define GPIO_Pin_7 ((uint32_t)0x0080)

Пин 7 выбран.

См. определение в файле niietcm4_gpio.h строка 326

6.47.2.19 #define GPIO_Pin_8 ((uint32_t)0x0100)

Пин 8 выбран.

См. определение в файле niietcm4_gpio.h строка 327

6.47.2.20 #define GPIO_Pin_8_11 ((uint32_t)0x0F00)

Пины 8-11 выбраны.

См. определение в файле niietcm4_gpio.h строка 337

6.47.2.21 #define GPIO_Pin_8_15 ((uint32_t)0xFF00)

Пины 8-15 выбраны.

См. определение в файле niietcm4_gpio.h строка 340

6.47.2.22 #define GPIO_Pin_9 ((uint32_t)0x0200)

Пин 9 выбран.

См. определение в файле niietcm4_gpio.h строка 328

6.47.2.23 `#define GPIO_Pin_All ((uint32_t)0xFFFF)`

Все пины выбраны.

См. определение в файле `niietcm4_gpio.h` строка 341

Используется в `GPIO_StructInit()`.

6.47.2.24 `#define IS_GET_GPIO_PIN(PIN)`

Макроопределение:

```
((PIN) == GPIO_Pin_0) || \
((PIN) == GPIO_Pin_1) || \
((PIN) == GPIO_Pin_2) || \
((PIN) == GPIO_Pin_3) || \
((PIN) == GPIO_Pin_4) || \
((PIN) == GPIO_Pin_5) || \
((PIN) == GPIO_Pin_6) || \
((PIN) == GPIO_Pin_7) || \
((PIN) == GPIO_Pin_8) || \
((PIN) == GPIO_Pin_9) || \
((PIN) == GPIO_Pin_10) || \
((PIN) == GPIO_Pin_11) || \
((PIN) == GPIO_Pin_12) || \
((PIN) == GPIO_Pin_13) || \
((PIN) == GPIO_Pin_14) || \
((PIN) == GPIO_Pin_15))
```

Макрос проверки номера пина при работе с пинами по отдельности.

См. определение в файле `niietcm4_gpio.h` строка 354

Используется в `GPIO_ReadBit()` и `GPIO_WriteBit()`.

6.48 Функции

Группы

- [Инициализация и деинициализация](#)
- [Чтение и запись](#)
- [Фильтрация](#)
- [Прерывания](#)

6.48.1 Подробное описание

6.49 Инициализация и деинициализация

Функции

- void [GPIO_DeInit](#) (NT_GPIO_TypeDef *GPIOx)
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- void [GPIO_Init](#) (NT_GPIO_TypeDef *GPIOx, [GPIO_Init_TypeDef](#) *GPIO_InitStruct)
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- void [GPIO_StructInit](#) ([GPIO_Init_TypeDef](#) *GPIO_InitStruct)
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.
- void [GPIO_AltFuncConfig](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_AltFunc_TypeDef](#) GPIO_AltFunc)

6.49.1 Подробное описание

6.49.2 Функции

6.49.2.1 void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)

Устанавливает все регистры выбранного GPIOx значениями по умолчанию.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Нет

См. определение в файле niitcm4_gpio.c строка 123

Перекрестные ссылки [GPIO_DATAOUT_Reset_Value](#), [GPIO_GPIODEN0_Reset_Value](#), [GPIO_GPIODEN1_Reset_Value](#), [GPIO_GPIODEN2_Reset_Value](#), [GPIO_GPIODEN3_Reset_Value](#), [GPIO_GPIOODCTLx_Reset_Value](#), [GPIO_GPIOODSCTLx_Reset_Value](#), [GPIO_GPIOPCTLx_Reset_Value](#), [GPIO_GPIOPUCTLx_Reset_Value](#), [GPIO_GPIOQEx_Reset_Value](#), [GPIO_GPIOQMx_Reset_Value](#), [GPIO_GPIOQPx_Reset_Value](#), [GPIO_GPIOSEx_Reset_Value](#), [GPIO_Regs_A_C_E_G_Mask](#), [GPIO_Regs_B_D_F_H_Mask](#) и [IS_GPIO_ALL_PERIPH](#).

6.49.2.2 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niitcm4_gpio.c строка 331

Перекрестные ссылки [GPIO_Init_TypeDef::GPIO_AltFunc](#), [GPIO_Init_TypeDef::GPIO_Dir](#), [GPIO_Dir_In](#), [GPIO_Dir_Out](#), [GPIO_Init_TypeDef::GPIO_Load](#), [GPIO_Load_16mA](#), [GPIO_Load_8mA](#), [GPIO_Init_TypeDef::GPIO_Mode](#), [GPIO_Mode_AltFunc](#), [GPIO_Mode_IO](#), [GPIO_Init_TypeDef::GPIO_Out](#), [GPIO_Out_Dis](#), [GPIO_Out_En](#), [GPIO_Init_TypeDef::GPIO_OutMode](#), [GPIO_OutMode_OD](#), [GPIO_OutMode_PP](#), [GPIO_Init_TypeDef::GPIO_Pin](#), [GPIO_Init_TypeDef::GPIO_PuPd](#)

_TypeDef::GPIO_PullUp, GPIO_PullUp_Dis, GPIO_PullUp_En, IS_GPIO_ALL_PERIPH, IS_GPIO_ALT_FUNC, IS_GPIO_DIR, IS_GPIO_LOAD, IS_GPIO_MODE, IS_GPIO_OUT, IS_GPIO_OUT_MODE, IS_GPIO_PIN и IS_GPIO_PULLUP.

Используется в RCC_SysClkDiv2Out().

6.49.2.3 void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)

Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет

См. определение в файле n1ietcm4_gpio.c строка 456

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_AltFunc_1, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_In, GPIO_Init_TypeDef::GPIO_Load, GPIO_Load_8mA, GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_IO, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_Dis, GPIO_Init_TypeDef::GPIO_OutMode, GPIO_OutMode_PP, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_All, GPIO_Init_TypeDef::GPIO_PullUp и GPIO_PullUp_Dis.

Используется в RCC_SysClkDiv2Out().

6.50 Чтение и запись

Группы

- [Чтение](#)
- [Запись](#)
- [Битовые операции](#)

6.50.1 Подробное описание

6.51 Чтение

Функции

- uint32_t [GPIO_ReadBit](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Чтение состояния выбранного пина.
- uint32_t [GPIO_Read](#) (NT_GPIO_TypeDef *GPIOx)
Чтение состояния выбранного порта GPIOx.
- uint32_t [GPIO_ReadMask](#) (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)
Чтение состояния выбранного порта GPIOx с использованием маски.

6.51.1 Подробное описание

6.51.2 Функции

6.51.2.1 uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx	
-----------------------	--

См. определение в файле niietcm4_gpio.c строка 503

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.51.2.2 uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина	
---------------------------	--

См. определение в файле niietcm4_gpio.c строка 477

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN и IS_GPIO_ALL_PERIPH.

6.51.2.3 uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)

Чтение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски чтения.

Возвращаемые значения

Состояние порта GPIOx с учетом маски	
---	--

См. определение в файле niietcm4_gpio.c строка 518

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.52 Запись

Функции

- void [GPIO_WriteBit](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [BitAction](#) BitVal)
Изменение состояния выбранного пина.
- void [GPIO_Write](#) (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx.
- void [GPIO_WriteMask](#) (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx с использованием маски.

6.52.1 Подробное описание

6.52.2 Функции

6.52.2.1 void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.52.2.2 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 546

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN, IS_GPIO_ALL_PERIPH и I↔S_GPIO_BIT_ACTION.

6.52.2.3 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 586

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

6.53 Битовые операции

Функции

- void **GPIO_SetBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Установка выбранных пинов.
- void **GPIO_ClearBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Сброс выбранных пинов.
- void **GPIO_ToggleBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Переключение выбранных пинов в противоположное состояние.

6.53.1 Подробное описание

6.53.2 Функции

6.53.2.1 void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 626

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.53.2.2 void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 609

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.53.2.3 void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Переключение выбранных пинов в противоположное состояние.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 643

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.54 Фильтрация

Функции

- void [GPIO_QualConfig](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_QualMode_TypeDef](#) Mode, uint32_t SamplePeriod)
Настройка фильтра выбранных пинов.
- void [GPIO_QualCmd](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение входных фильтров.
- void [GPIO_SyncCmd](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение пересинхронизации входов через 2 триггера-защелки.

6.54.1 Подробное описание

6.54.2 Функции

6.54.2.1 void [GPIO_QualCmd](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.54.2.2 void [GPIO_QualConfig](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [GPIO_QualMode_TypeDef](#) Mode, uint32_t SamplePeriod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 664

Перекрестные ссылки GPIO_QualMode_3sample, GPIO_QualMode_6sample, IS_GPIO_ALL_PERIPH, IS_GPIO_PIN, IS_GPIO_QUAL_MODE и IS_GPIO_QUAL_PERIOD.

6.54.2.3 void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 814

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.55 Прерывания

Функции

- void [GPIO_ITConfig](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_IntType_TypeDef](#) IntType, [GPIO_IntPol_TypeDef](#) IntPol)
Настройка прерываний пинов.
- void [GPIO_ITCmd](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение прерываний выбранных пинов.
- void [GPIO_ITStatusClear](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Очистка флагов прерываний выбранных пинов.

6.55.1 Подробное описание

6.55.2 Функции

6.55.2.1 void [GPIO_ITCmd](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 913

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.55.2.2 void [GPIO_ITConfig](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [GPIO_IntType_TypeDef](#) IntType, [GPIO_IntPol_TypeDef](#) IntPol)

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 876

Перекрестные ссылки [GPIO_IntPol_Neg](#), [GPIO_IntPol_Pos](#), [GPIO_IntType_Edge](#), [GPIO_IntType_Level](#), [IS_GPIO_ALL_PERIPH](#), [IS_GPIO_INT_POL](#), [IS_GPIO_INT_TYPE](#) и [IS_GPIO_PIN](#).

6.55.2.3 void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле n1ietcm4_gpio.c строка 938

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.56 Константы

Макросы

- `#define RCC_CLK_CHANGE_TIMEOUT ((uint32_t)10000)`
- `#define RCC_CLK_PLL_STABLE_TIMEOUT ((uint32_t)100)`

6.56.1 Подробное описание

6.56.2 Макросы

6.56.2.1 `#define RCC_CLK_CHANGE_TIMEOUT ((uint32_t)10000)`

Время ожидания смены источника тактирования

См. определение в файле `niietcm4_rcc.h` строка 52

Используется в `RCC_WaitClkChange()`.

6.56.2.2 `#define RCC_CLK_PLL_STABLE_TIMEOUT ((uint32_t)100)`

Время ожидания стабилизации выходной частоты PLL

См. определение в файле `niietcm4_rcc.h` строка 53

Используется в `RCC_PLLAutoConfig()`.

6.57 Типы

Структуры данных

- struct [RCC_PLLInit_TypeDef](#)
Структура инициализации PLL.

Макросы

- #define [IS_RCC_PLL_REF](#)(PLL_REF)
Макрос проверки аргументов типа [RCC_PLLRef_TypeDef](#).
- #define [IS_RCC_PLL_NO](#)(PLL_NO)
Макрос проверки аргументов типа [RCC_PLLNO_TypeDef](#).
- #define [IS_RCC_UART_CLK](#)(UART_CLK)
Макрос проверки аргументов типа [RCC_UARTClk_TypeDef](#).
- #define [IS_RCC_SPI_CLK](#)(SPI_CLK)
Макрос проверки аргументов типа [RCC_SPIClk_TypeDef](#).
- #define [IS_RCC_USB_CLK](#)(USB_CLK)
Макрос проверки аргументов типа [RCC_USBClk_TypeDef](#).
- #define [IS_RCC_USB_FREQ](#)(USB_FREQ)
Макрос проверки аргументов типа [RCC_USBFreq_TypeDef](#).
- #define [IS_RCC_ADC_CLK](#)(ADC_CLK)
Макрос проверки аргументов типа [RCC_ADCClk_TypeDef](#).
- #define [IS_RCC_SYS_CLK](#)(SYS_CLK)
Макрос проверки аргументов типа [RCC_SysClk_TypeDef](#).
- #define [IS_RCC_PERIPH_CLK](#)(PERIPH_CLK)
Макрос проверки аргументов типа [RCC_PeriphClk_TypeDef](#).
- #define [IS_RCC_PERIPH_RST](#)(PERIPH_RST)
Макрос проверки аргументов типа [RCC_PeriphRst_TypeDef](#).
- #define [IS_RCC_PLLDIV](#)(PLLDIV) (((PLLDIV) & ((uint32_t)0xFFFFF00)) == ((uint32_t)0x00))
Макрос проверки значения выходного делителя PLL на попадание в допустимый диапазон.
- #define [IS_RCC_PLL_NR](#)(PLL_NR) (((PLL_NR) <= ((uint32_t)33)) && ((PLL_NR) >= ((uint32_t)2)))
Макрос проверки значения опорного делителя PLL на попадание в допустимый диапазон.
- #define [IS_RCC_PLL_NF](#)(PLL_NF) (((PLL_NF) <= ((uint32_t)513)) && ((PLL_NF) >= ((uint32_t)2)))
Макрос проверки значения делителя ОС PLL на попадание в допустимый диапазон.
- #define [IS_RCC_CLK_DIV](#)(CLK_DIV) ((CLK_DIV) < ((uint32_t)64))
Макрос проверки значения делителя тактового сигнала на попадание в допустимый диапазон.
- #define [IS_RCC_SYS_FREQ](#)(SYS_FREQ) (((SYS_FREQ) < ((uint32_t)200000000)) && ((SYS_FREQ) >= ((uint32_t)1000000)))
Макрос проверки значения желаемой частоты при автонастройке в допустимый диапазон.

Перечисления

- enum [RCC_PLLRef_TypeDef](#) { [RCC_PLLRef_XI_OSC](#), [RCC_PLLRef_USB_CLK](#), [RCC_PLLRef_USB_60MHz](#), [RCC_PLLRef_ETH_25MHz](#) }
Выбор источника опорного сигнала PLL.
- enum [RCC_PLLNO_TypeDef](#) { [RCC_PLLNO_Disable](#), [RCC_PLLNO_Div2](#), [RCC_PLLNO_Div4](#) = 3 }

Выходной делитель NO.

- enum `RCC_UARTClk_TypeDef` { `RCC_UARTClk_SYSClk`, `RCC_UARTClk_XI_OSC`, `RCC_UARTClk_USB_Clk`, `RCC_UARTClk_USB_60MHz` }

Выбор источника тактирования для UART.

- enum `RCC_SPIClk_TypeDef` { `RCC_SPIClk_SYSClk`, `RCC_SPIClk_XI_OSC`, `RCC_SPIClk_USB_Clk`, `RCC_SPIClk_USB_60MHz` }

Выбор источника тактирования для SPI.

- enum `RCC_USBClk_TypeDef` { `RCC_USBClk_XI_OSC`, `RCC_USBClk_USB_Clk` }

Выбор источника тактирования для USB.

- enum `RCC_USBFreq_TypeDef` { `RCC_USBFreq_12MHz`, `RCC_USBFreq_24MHz` }

Выбор фиксированной частоты на входе CLK_USB.

- enum `RCC_ADCClk_TypeDef` { `RCC_ADCClk_0`, `RCC_ADCClk_1`, `RCC_ADCClk_2`, `RCC_ADCClk_3`, `RCC_ADCClk_4`, `RCC_ADCClk_5`, `RCC_ADCClk_6`, `RCC_ADCClk_7`, `RCC_ADCClk_8`, `RCC_ADCClk_9`, `RCC_ADCClk_10`, `RCC_ADCClk_11` }

Выбор модуля ADC для настройки его тактового сигнала.

- enum `RCC_SysClk_TypeDef` { `RCC_SysClk_CPE_Sel`, `RCC_SysClk_POR`, `RCC_SysClk_XI_OSC`, `RCC_SysClk_PLL`, `RCC_SysClk_PLLDIV`, `RCC_SysClk_USB60MHz`, `RCC_SysClk_USB_Clk`, `RCC_SysClk_ETH25MHz` }

Выбор источника системной частоты.

- enum `RCC_PeriphClk_TypeDef` { `RCC_PeriphClk_QEP0` = ((uint32_t)(1<<1)), `RCC_PeriphClk_QEP1` = ((uint32_t)(1<<2)), `RCC_PeriphClk_CMP` = ((uint32_t)(1<<9)), `RCC_PeriphClk_PWM0` = ((uint32_t)(1<<10)), `RCC_PeriphClk_PWM1` = ((uint32_t)(1<<11)), `RCC_PeriphClk_PWM2` = ((uint32_t)(1<<12)), `RCC_PeriphClk_PWM3` = ((uint32_t)(1<<13)), `RCC_PeriphClk_PWM4` = ((uint32_t)(1<<14)), `RCC_PeriphClk_PWM5` = ((uint32_t)(1<<15)), `RCC_PeriphClk_PWM6` = ((uint32_t)(1<<16)), `RCC_PeriphClk_PWM7` = ((uint32_t)(1<<17)), `RCC_PeriphClk_PWM8` = ((uint32_t)(1<<18)), `RCC_PeriphClk_WD` = ((uint32_t)(1<<19)), `RCC_PeriphClk_I2C0` = ((uint32_t)(1<<20)), `RCC_PeriphClk_I2C1` = ((uint32_t)(1<<21)), `RCC_PeriphClk_ADC` = ((uint32_t)(1<<24)) }

Управление тактированием периферийных блоков

- enum `RCC_PeriphRst_TypeDef` { `RCC_PeriphRst_WD`, `RCC_PeriphRst_I2C0`, `RCC_PeriphRst_I2C1`, `RCC_PeriphRst_USB`, `RCC_PeriphRst_Timer0`, `RCC_PeriphRst_Timer1`, `RCC_PeriphRst_Timer2`, `RCC_PeriphRst_UART0`, `RCC_PeriphRst_UART1`, `RCC_PeriphRst_UART2`, `RCC_PeriphRst_UART3`, `RCC_PeriphRst_SPI0`, `RCC_PeriphRst_SPI1`, `RCC_PeriphRst_SPI2`, `RCC_PeriphRst_SPI3`, `RCC_PeriphRst_ETH`, `RCC_PeriphRst_QEP0`, `RCC_PeriphRst_QEP1`, `RCC_PeriphRst_PWM0`, `RCC_PeriphRst_PWM1`, `RCC_PeriphRst_PWM2`, `RCC_PeriphRst_PWM3`, `RCC_PeriphRst_PWM4`, `RCC_PeriphRst_PWM5`, `RCC_PeriphRst_PWM6`, `RCC_PeriphRst_PWM7`, `RCC_PeriphRst_PWM8`, `RCC_PeriphRst_CAP0`, `RCC_PeriphRst_CAP1`, `RCC_PeriphRst_CAP2`, `RCC_PeriphRst_CAP3`, `RCC_PeriphRst_CAP4`, `RCC_PeriphRst_CAP5`, `RCC_PeriphRst_CMP` }

Управление сбросом периферийных блоков

6.57.1 Подробное описание

6.57.2 Макросы

6.57.2.1 #define IS_RCC_ADC_CLK(ADC_CLK)

Макроопределение:

```
((ADC_CLK) == RCC_ADCClk_0) || \
((ADC_CLK) == RCC_ADCClk_1) || \
((ADC_CLK) == RCC_ADCClk_2) || \
((ADC_CLK) == RCC_ADCClk_3) || \
((ADC_CLK) == RCC_ADCClk_4) || \
((ADC_CLK) == RCC_ADCClk_5) || \
((ADC_CLK) == RCC_ADCClk_6) || \
((ADC_CLK) == RCC_ADCClk_7) || \
((ADC_CLK) == RCC_ADCClk_8) || \
((ADC_CLK) == RCC_ADCClk_9) || \
((ADC_CLK) == RCC_ADCClk_10) || \
((ADC_CLK) == RCC_ADCClk_11)
```

Макрос проверки аргументов типа [RCC_ADCClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 204

Используется в `RCC_ADCClkCmd()` и `RCC_ADCClkDivConfig()`.

6.57.2.2 #define IS_RCC_PERIPH_CLK(PERIPH_CLK)

Макроопределение:

```
((PERIPH_CLK) == RCC_PeriphClk_QEP0) || \
((PERIPH_CLK) == RCC_PeriphClk_QEP1) || \
((PERIPH_CLK) == RCC_PeriphClk_CMP) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM0) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM1) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM2) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM4) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM5) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM6) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM7) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM8) || \
((PERIPH_CLK) == RCC_PeriphClk_WD) || \
((PERIPH_CLK) == RCC_PeriphClk_I2C0) || \
((PERIPH_CLK) == RCC_PeriphClk_I2C1) || \
((PERIPH_CLK) == RCC_PeriphClk_ADC)
```

Макрос проверки аргументов типа [RCC_PeriphClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 274

Используется в `RCC_PeriphClkCmd()`.

6.57.2.3 #define IS_RCC_PLL_NO(PLL_NO)

Макроопределение:

```
((PLL_NO) == RCC_PLLNO_Disable) || \
((PLL_NO) == RCC_PLLNO_Div2) || \
((PLL_NO) == RCC_PLLNO_Div4)
```

Макрос проверки аргументов типа [RCC_PLLNO_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 100

Используется в `RCC_PLLInit()`.

6.57.2.4 #define IS_RCC_PLL_REF(PLL_REF)

Макроопределение:

```
((PLL_REF) == RCC_PLLRef_XI_OSC) || \
    ((PLL_REF) == RCC_PLLRef_USB_CLK) || \
    ((PLL_REF) == RCC_PLLRef_USB_60MHz) || \
    ((PLL_REF) == RCC_PLLRef_ETH_25MHz))
```

Макрос проверки аргументов типа [RCC_PLLRef_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 79

Используется в `RCC_PLLAutoConfig()` и `RCC_PLLInit()`.

6.57.2.5 `#define IS_RCC_SPI_CLK(SPI_CLK)`

Макроопределение:

```
((SPI_CLK) == RCC_SPIClk_SYSCLK) || \
    ((SPI_CLK) == RCC_SPIClk_XI_OSC) || \
    ((SPI_CLK) == RCC_SPIClk_USB_CLK) || \
    ((SPI_CLK) == RCC_SPIClk_USB_60MHz))
```

Макрос проверки аргументов типа [RCC_SPIClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 141

Используется в `RCC_SPIClkSel()`.

6.57.2.6 `#define IS_RCC_SYS_CLK(SYS_CLK)`

Макроопределение:

```
((SYS_CLK) == RCC_SysClk_CPE_Sel) || \
    ((SYS_CLK) == RCC_SysClk_POR) || \
    ((SYS_CLK) == RCC_SysClk_XI_OSC) || \
    ((SYS_CLK) == RCC_SysClk_PL) || \
    ((SYS_CLK) == RCC_SysClk_PLLDIV) || \
    ((SYS_CLK) == RCC_SysClk_USB60MHz) || \
    ((SYS_CLK) == RCC_SysClk_USB_CLK) || \
    ((SYS_CLK) == RCC_SysClk_ETH25MHz))
```

Макрос проверки аргументов типа [RCC_SysClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 237

Используется в `RCC_SysClkSel()`.

6.57.2.7 `#define IS_RCC_UART_CLK(UART_CLK)`

Макроопределение:

```
((UART_CLK) == RCC_UARTClk_SYSCLK) || \
    ((UART_CLK) == RCC_UARTClk_XI_OSC) || \
    ((UART_CLK) == RCC_UARTClk_USB_CLK) || \
    ((UART_CLK) == RCC_UARTClk_USB_60MHz))
```

Макрос проверки аргументов типа [RCC_UARTClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 120

Используется в `RCC_UARTClkSel()`.

6.57.2.8 `#define IS_RCC_USB_CLK(USB_CLK)`

Макроопределение:

```
((USB_CLK) == RCC_USBClk_XI_OSC) || \
    ((USB_CLK) == RCC_USBClk_USB_CLK))
```

Макрос проверки аргументов типа [RCC_USBCLK_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 160

Используется в `RCC_USBCLKConfig()`.

6.57.2.9 `#define IS_RCC_USB_FREQ(USB_FREQ)`

Макроопределение:

```
((USB_FREQ) == RCC_USBFreq_12MHz) || \
((USB_FREQ) == RCC_USBFreq_24MHz))
```

Макрос проверки аргументов типа [RCC_USBFreq_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 177

Используется в `RCC_USBCLKConfig()`.

6.57.3 Перечисления

6.57.3.1 `enum RCC_ADCClk_TypeDef`

Выбор модуля ADC для настройки его тактового сигнала.

Элементы перечислений

<code>RCC_ADCClk_0</code>	Тактовый сигнал ADC 0
<code>RCC_ADCClk_1</code>	Тактовый сигнал ADC 1
<code>RCC_ADCClk_2</code>	Тактовый сигнал ADC 2
<code>RCC_ADCClk_3</code>	Тактовый сигнал ADC 3
<code>RCC_ADCClk_4</code>	Тактовый сигнал ADC 4
<code>RCC_ADCClk_5</code>	Тактовый сигнал ADC 5
<code>RCC_ADCClk_6</code>	Тактовый сигнал ADC 6
<code>RCC_ADCClk_7</code>	Тактовый сигнал ADC 7
<code>RCC_ADCClk_8</code>	Тактовый сигнал ADC 8
<code>RCC_ADCClk_9</code>	Тактовый сигнал ADC 9
<code>RCC_ADCClk_10</code>	Тактовый сигнал ADC 10
<code>RCC_ADCClk_11</code>	Тактовый сигнал ADC 11

См. определение в файле `niietcm4_rcc.h` строка 184

6.57.3.2 `enum RCC_PeriphClk_TypeDef`

Управление тактированием периферийных блоков

Элементы перечислений

<code>RCC_PeriphClk_QEP0</code>	Управление тактированием блока QEP 0
<code>RCC_PeriphClk_QEP1</code>	Управление тактированием блока QEP 1
<code>RCC_PeriphClk_CMP</code>	Управление тактированием блока аналогового компаратора
<code>RCC_PeriphClk_PWM0</code>	Управление тактированием блока PWM 0
<code>RCC_PeriphClk_PWM1</code>	Управление тактированием блока PWM 1
<code>RCC_PeriphClk_PWM2</code>	Управление тактированием блока PWM 2

RCC_PeriphClk_PWM3 Управление тактированием блока PWM 3
 RCC_PeriphClk_PWM4 Управление тактированием блока PWM 4
 RCC_PeriphClk_PWM5 Управление тактированием блока PWM 5
 RCC_PeriphClk_PWM6 Управление тактированием блока PWM 6
 RCC_PeriphClk_PWM7 Управление тактированием блока PWM 7
 RCC_PeriphClk_PWM8 Управление тактированием блока PWM 8
 RCC_PeriphClk_WD Управление тактированием сторожевого таймера
 RCC_PeriphClk_I2C0 Управление тактированием блока I2C 0
 RCC_PeriphClk_I2C1 Управление тактированием блока I2C 1
 RCC_PeriphClk_ADC Управление тактированием контроллера ADC

См. определение в файле niietcm4_rcc.h строка 250

6.57.3.3 enum RCC_PeriphRst_TypeDef

Управление сбросом периферийных блоков

Элементы перечислений

RCC_PeriphRst_WD Управление сбросом сторожевого таймера
 RCC_PeriphRst_I2C0 Управление сбросом блока I2C 0
 RCC_PeriphRst_I2C1 Управление сбросом блока I2C 1
 RCC_PeriphRst_USB Управление сбросом блока USB
 RCC_PeriphRst_Timer0 Управление сбросом блока Timer 0
 RCC_PeriphRst_Timer1 Управление сбросом блока Timer 1
 RCC_PeriphRst_Timer2 Управление сбросом блока Timer 2
 RCC_PeriphRst_UART0 Управление сбросом блока UART 0
 RCC_PeriphRst_UART1 Управление сбросом блока UART 1
 RCC_PeriphRst_UART2 Управление сбросом блока UART 2
 RCC_PeriphRst_UART3 Управление сбросом блока UART 3
 RCC_PeriphRst_SPI0 Управление сбросом блока SPI 0
 RCC_PeriphRst_SPI1 Управление сбросом блока SPI 1
 RCC_PeriphRst_SPI2 Управление сбросом блока SPI 2
 RCC_PeriphRst_SPI3 Управление сбросом блока SPI 3
 RCC_PeriphRst_ETH Управление сбросом блока Ethernet
 RCC_PeriphRst_QEP0 Управление сбросом блока QEP 0
 RCC_PeriphRst_QEP1 Управление сбросом блока QEP 1
 RCC_PeriphRst_PWM0 Управление сбросом блока PWM 0
 RCC_PeriphRst_PWM1 Управление сбросом блока PWM 1
 RCC_PeriphRst_PWM2 Управление сбросом блока PWM 2
 RCC_PeriphRst_PWM3 Управление сбросом блока PWM 3
 RCC_PeriphRst_PWM4 Управление сбросом блока PWM 4
 RCC_PeriphRst_PWM5 Управление сбросом блока PWM 5
 RCC_PeriphRst_PWM6 Управление сбросом блока PWM 6
 RCC_PeriphRst_PWM7 Управление сбросом блока PWM 7
 RCC_PeriphRst_PWM8 Управление сбросом блока PWM 8
 RCC_PeriphRst_CAP0 Управление сбросом блока CAP 0

RCC_PeriphRst_CAP1 Управление сбросом блока CAP 1
RCC_PeriphRst_CAP2 Управление сбросом блока CAP 2
RCC_PeriphRst_CAP3 Управление сбросом блока CAP 3
RCC_PeriphRst_CAP4 Управление сбросом блока CAP 4
RCC_PeriphRst_CAP5 Управление сбросом блока CAP 5
RCC_PeriphRst_CMP Управление сбросом блока аналогового компаратора

См. определение в файле niietcm4_rcc.h строка 294

6.57.3.4 enum RCC_PLLNO_TypeDef

Выходной делитель NO.

Элементы перечислений

RCC_PLLNO_Disable Делитель NO выключен
RCC_PLLNO_Div2 Коэффициент деления NO равен 2
RCC_PLLNO_Div4 Коэффициент деления NO равен 4

См. определение в файле niietcm4_rcc.h строка 89

6.57.3.5 enum RCC_PLLRef_TypeDef

Выбор источника опорного сигнала PLL.

Элементы перечислений

RCC_PLLRef_XI_OSC Сигнал со входа XI_OSC
RCC_PLLRef_USB_CLK Сигнал с входной альтернативной функции CLK_USB
RCC_PLLRef_USB_60MHz Сигнал на выходе блока USB
RCC_PLLRef_ETH_25MHz Входной тактовый сигнал блока Ethernet

См. определение в файле niietcm4_rcc.h строка 67

6.57.3.6 enum RCC_SPIClk_TypeDef

Выбор источника тактирования для SPI.

Элементы перечислений

RCC_SPIClk_SYSCLK Текущая системная частота
RCC_SPIClk_XI_OSC Сигнал со входа XI_OSC
RCC_SPIClk_USB_CLK Сигнал с входной альтернативной функции CLK_USB
RCC_SPIClk_USB_60MHz Сигнал на выходе блока USB

См. определение в файле niietcm4_rcc.h строка 129

6.57.3.7 enum RCC_SysClk_TypeDef

Выбор источника системной частоты.

Элементы перечислений

RCC_SysClk_CPE_Sel Источник определяется состоянием вывода CPE: 0-POR, 1-XI_OSC

RCC_SysClk_POR Внутренний источник тактового сигнала

RCC_SysClk_XI_OSC Внешний источник тактового сигнала на входе XI_OSC

RCC_SysClk_PLL Выход блока PLL

RCC_SysClk_PLLDIV Выход блока PLL через делитель PLL DIV

RCC_SysClk_USB60MHz Выход блока USB 60 МГц

RCC_SysClk_USB_CLK Внешний источник тактового сигнала на входе CLK_USB

RCC_SysClk_ETH25MHz Входной тактовый сигнал блока Ethernet

См. определение в файле niietcm4_rcc.h строка 221

6.57.3.8 enum RCC_UARTClk_TypeDef

Выбор источника тактирования для UART.

Элементы перечислений

RCC_UARTClk_SYSCLK Текущая системная частота

RCC_UARTClk_XI_OSC Сигнал со входа XI_OSC

RCC_UARTClk_USB_CLK Сигнал с входной альтернативной функции CLK_USB

RCC_UARTClk_USB_60MHz Сигнал на выходе блока USB

См. определение в файле niietcm4_rcc.h строка 108

6.57.3.9 enum RCC_USBClk_TypeDef

Выбор источника тактирования для USB.

Элементы перечислений

RCC_USBClk_XI_OSC Сигнал со входа XI_OSC

RCC_USBClk_USB_CLK Сигнал с входной альтернативной функции CLK_USB

См. определение в файле niietcm4_rcc.h строка 150

6.57.3.10 enum RCC_USBFreq_TypeDef

Выбор фиксированной частоты на входе CLK_USB.

Элементы перечислений

RCC_USBFreq_12MHz 12 МГц сигнал на входе CLK_USB

RCC_USBFreq_24MHz 24 МГц сигнал на входе CLK_USB

См. определение в файле niietcm4_rcc.h строка 167

6.58 Функции

Группы

- [Конфигурация PLL](#)
- [Управление тактированием](#)
- [Управление сбросом](#)

Функции

- void [RCC_SysClkDiv2Out](#) ([FunctionalState](#) State)

Включение генерации тактового сигнала с частой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

6.58.1 Подробное описание

6.58.2 Функции

6.58.2.1 void [RCC_SysClkDiv2Out](#) ([FunctionalState](#) State)

Включение генерации тактового сигнала с частой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState : <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 106

Перекрестные ссылки [GPIO_InitTypeDef::GPIO_AltFunc](#), [GPIO_AltFunc_2](#), [GPIO_InitTypeDef::GPIO_Dir](#), [GPIO_Dir_Out](#), [GPIO_Init\(\)](#), [GPIO_InitTypeDef::GPIO_Mode](#), [GPIO_Mode_AltFunc](#), [GPIO_InitTypeDef::GPIO_Out](#), [GPIO_Out_En](#), [GPIO_InitTypeDef::GPIO_Pin](#), [GPIO_Pin_0](#), [GPIO_StructInit\(\)](#) и [IS_FUNCTIONAL_STATE](#).

6.59 Конфигурация PLL

Функции

- `OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)`
Автоматическая конфигурация PLL для получения желаемой системной частоты.
- `void RCC_PLLInit (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)`
Инициализирует PLL согласно параметрам структуры `RCC_PLLInit_Struct`.
- `void RCC_PLLDeInit ()`
Устанавливает все регистры PLL значениями по умолчанию.
- `void RCC_PLLStructInit (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)`
Заполнение каждого члена структуры `RCC_PLLInit_Struct` значениями по умолчанию.
- `void RCC_PLLPowerDownCmd (FunctionalState State)`
Управление режимом PowerDown PLL.

6.59.1 Подробное описание

6.59.2 Функции

6.59.2.1 `OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)`

Автоматическая конфигурация PLL для получения желаемой системной частоты.

С учетом данных об источнике частоты для PLL, а также о значении желаемой частоты, вычисляются все необходимые коэффициенты.

Внимание

Если $\text{Freq} < 50$ МГц, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

RCC_PLLRef	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из <code>RCC_PLLRef_TypeDef</code> .
SysFreq	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 142

Перекрестные ссылки `EXT_OSC_VALUE`, `IS_RCC_PLL_REF`, `IS_RCC_SYS_FREQ`, `RCC_CLK_PLL_STABLE_TIMEOUT`, `RCC_PLLInit_TypeDef::RCC_PLLDiv`, `RCC_PLLInit()`, `RCC_PLLInit_TypeDef::RCC_PLLNF`, `RCC_PLLInit_TypeDef::RCC_PLLNO`, `RCC_PLLNO_Div2`, `RCC_PLLNO_Div4`, `RCC_PLLInit_TypeDef::RCC_PLLNR`, `RCC_PLLInit_TypeDef::RCC_PLLRef`, `RCC_PLLRef_ETH_25MHz`, `RCC_PLLRef_USB_60MHz`, `RCC_PLLRef_USB_CLK`, `RCC_PLLRef_XI_OSC`, `RCC_SysClk_PLL`, `RCC_SysClk_PLLDIV` и `RCC_SysClkSel()`.

6.59.2.2 `void RCC_PLLDeInit ()`

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 298

Перекрестные ссылки `RCC_PLL_CTRL_Reset_Value`, `RCC_PLL_NF_Reset_Value`, `RCC_PLLNR_Reset_Value` и `RCC_PLL_OD_Reset_Value`.

6.59.2.3 void `RCC_PLLInit` (`RCC_PLLInit_TypeDef` * `RCC_PLLInit_Struct`)

Инициализирует PLL согласно параметрам структуры `RCC_PLLInit_Struct`.

Значение выходной частоты PLL вычисляется с использованием значений опорного NR и выходного NO делителей, а также делителя обратной связи NF по формуле:

$$F_{OUT} = (F_{IN} \times NF) / (NO \times NR),$$

где F_{IN} – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- $3,2 \text{ МГц} < F_{IN} < 150 \text{ МГц}$,
- $800 \text{ КГц} < F_{REF} < 8 \text{ МГц}$,
- $200 \text{ МГц} < F_{VCO} < 500 \text{ МГц}$,

где частота фазового детектора F_{REF} вычисляется по формуле:

$$F_{REF} = F_{IN} / (2 \times NR),$$

а частота F_{VCO} вычисляется по формуле:

$$F_{VCO} = F_{IN} \times (NF / NR)$$

Аргументы

<code>RCC_PLLInit_Struct</code>	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
---------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 262

Перекрестные ссылки `IS_RCC_PLL_NF`, `IS_RCC_PLL_NO`, `IS_RCC_PLL_NR`, `IS_RCC_PLL_REF`, `IS_RCC_PLLDIV`, `RCC_PLLInit_TypeDef::RCC_PLLDiv`, `RCC_PLLInit_TypeDef::RCC_PLLNF`, `RCC_PLLInit_TypeDef::RCC_PLLNO`, `RCC_PLLInit_TypeDef::RCC_PLLNR` и `RCC_PLLInit_TypeDef::RCC_PLLRef`.

Используется в `RCC_PLLAutoConfig()`.

6.59.2.4 void RCC_PLLPowerDownCmd (FunctionalState State)

Управление режимом PowerDown PLL.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 313

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.59.2.5 void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 284

Перекрестные ссылки RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit_TypeDef::RCC_PL↔LNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLNO_Disable, RCC_PLLInit_TypeDef::R↔CC_PLLNR, RCC_PLLInit_TypeDef::RCC_PLLRef и RCC_PLLRef_XI_OSC.

6.60 Управление тактированием

Группы

- [Тактирование USB](#)
- [Тактирование UART](#)
- [Тактирование SPI](#)
- [Тактирование ADC](#)

Функции

- `void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)`
Включение тактирования выбранного блока периферии.
- `OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)`
Выбор источника для системного тактового сигнала.
- `RCC_SysClk_TypeDef RCC_SysClkStatus ()`
Текущий источник системного тактового сигнала.

6.60.1 Подробное описание

6.60.2 Функции

6.60.2.1 `void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)`

Включение тактирования выбранного блока периферии.

Внимание

Блоки UART , SPI, ADC, USB управляются отдельно.

- [Тактирование UART](#)
- [Тактирование SPI](#)
- [Тактирование ADC](#)
- [Тактирование USB](#)

Аргументы

RCC_PeriphClk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 342

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_RCC_PERIPH_CLK`.

6.60.2.2 `OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)`

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_TypeDef .
------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 365

Перекрестные ссылки `IS_RCC_SYS_CLK` и `RCC_WaitClkChange()`.

Используется в `RCC_PLLAutoConfig()`.

6.60.2.3 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле `niietcm4_rcc.c` строка 394

6.61 Тактирование USB

Функции

- void [RCC_USBCLKConfig](#) ([RCC_USBCLK_TypeDef](#) RCC_USBCLK, [RCC_USBFreq_TypeDef](#) RCC_USBFreq)
Настройка источника тактового сигнала для USB.
- void [RCC_USBCLKCmd](#) ([FunctionalState](#) State)
Включение тактирования USB.

6.61.1 Подробное описание

6.61.2 Функции

6.61.2.1 void RCC_USBCLKCmd (FunctionalState State)

Включение тактирования USB.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 425

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

6.61.2.2 void RCC_USBCLKConfig (RCC_USBCLK_TypeDef RCC_USBCLK, RCC_USBFreq_TypeDef RCC_USBFreq)

Настройка источника тактового сигнала для USB.

Аргументы

RCC_USBCLK	Выбор источника тактирования. Параметр принимает любое значение из RCC_USBCLK_TypeDef .
RCC_USBFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 408

Перекрестные ссылки `IS_RCC_USB_CLK` и `IS_RCC_USB_FREQ`.

6.62 Тактирование UART

Функции

- void [RCC_UARTClkSel](#) (NT_UART_TypeDef *UARTx, [RCC_UARTClk_TypeDef](#) RCC_UARTClk)
Настройка источника тактового сигнала для выбранного UART.
- void [RCC_UARTClkDivConfig](#) (NT_UART_TypeDef *UARTx, uint32_t DivVal, [FunctionalState](#) DivState)
Настройка делителя тактового сигнала для выбранного UART.
- void [RCC_UARTClkCmd](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Включение тактирования UART.

6.62.1 Подробное описание

6.62.2 Функции

6.62.2.1 void [RCC_UARTClkCmd](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 526

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.62.2.2 void [RCC_UARTClkDivConfig](#) (NT_UART_TypeDef * UARTx, uint32_t DivVal, [FunctionalState](#) DivState)

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 488

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_UART_ALL_PERIPH.

6.62.2.3 void RCC_UARTClkSel (NT_UART_TypeDef * UARTx, RCC_UARTClk_TypeDef
RCC_UARTClk)

Настройка источника тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
RCC_UART↔ Clk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 448

Перекрестные ссылки `IS_RCC_UART_CLK` и `IS_UART_ALL_PERIPH`.

6.63 Тактирование SPI

Функции

- void [RCC_SPIClkSel](#) (NT_SPI_TypeDef *SPIx, [RCC_SPIClk_TypeDef](#) RCC_SPIClk)
Настройка источника тактового сигнала для выбранного SPI.
- void [RCC_SPIClkDivConfig](#) (NT_SPI_TypeDef *SPIx, uint32_t DivVal, [FunctionalState](#) DivState)
Настройка делителя тактового сигнала для выбранного SPI.
- void [RCC_SPIClkCmd](#) (NT_SPI_TypeDef *SPIx, [FunctionalState](#) State)
Включение тактирования SPI.

6.63.1 Подробное описание

6.63.2 Функции

6.63.2.1 void [RCC_SPIClkCmd](#) (NT_SPI_TypeDef * SPIx, [FunctionalState](#) State)

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 648

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_SPI_ALL_PERIPH`.

6.63.2.2 void [RCC_SPIClkDivConfig](#) (NT_SPI_TypeDef * SPIx, uint32_t DivVal, [FunctionalState](#) DivState)

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 610

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_SPI_ALL_PERIPH`.

6.63.2.3 void [RCC_SPIClkSel](#) (NT_SPI_TypeDef * SPIx, [RCC_SPIClk_TypeDef](#) RCC_SPIClk)

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 569

Перекрестные ссылки `IS_RCC_SPI_CLK` и `IS_SPI_ALL_PERIPH`.

6.64 Тактирование ADC

Функции

- void [RCC_ADCClkDivConfig](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, uint32_t DivVal, [FunctionalState](#) DivState)
Настройка делителя тактового сигнала для выбранного ADC.
- void [RCC_ADCClkCmd](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, [FunctionalState](#) State)
Включение тактирования ADC.

6.64.1 Подробное описание

6.64.2 Функции

6.64.2.1 void [RCC_ADCClkCmd](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, [FunctionalState](#) State)

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 779

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_ADC_CLK, RCC_ADCClk_0, RCC_ADCClk_1, RCC_ADCClk_10, RCC_ADCClk_2, RCC_ADCClk_3, RCC_ADCClk_4, RCC_ADCClk_5, RCC_ADCClk_6, RCC_ADCClk_7, RCC_ADCClk_8 и RCC_ADCClk_9.

6.64.2.2 void [RCC_ADCClkDivConfig](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, uint32_t DivVal, [FunctionalState](#) DivState)

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 695

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_ADC_CLK, IS_RCC_CLK_DIV, RCC_ADCClk_0, RCC_ADCClk_1, RCC_ADCClk_10, RCC_ADCClk_2, RCC_ADCClk_3, RCC_ADCClk_4, RCC_ADCClk_5, RCC_ADCClk_6, RCC_ADCClk_7, RCC_ADCClk_8 и RCC_ADCClk_9.

6.65 Управление сбросом

Функции

- void [RCC_PeriphRstCmd](#) ([RCC_PeriphRst_TypeDef](#) RCC_PeriphRst, [FunctionalState](#) State)
Вывод из состояния сброса периферийных блоков.

6.65.1 Подробное описание

6.65.2 Функции

6.65.2.1 void [RCC_PeriphRstCmd](#) ([RCC_PeriphRst_TypeDef](#) RCC_PeriphRst, [FunctionalState](#) State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_PeriphRst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 869

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_RCC_PERIPH_RST](#) и [RCC_PeriphRst_ETH](#).

Используется в [CAP_DeInit\(\)](#) и [UART_DeInit\(\)](#).

6.66 Типы

Структуры данных

- struct `RTC_Time_TypeDef`
Структура времени.
- struct `RTC_Date_TypeDef`
Структура даты.

Макросы

- `#define IS_RTC_PSECOND(PSECOND) ((PSECOND) <= 0x3FF)`
Макрос проверки попадания значений долей секунд в допустимый диапазон.
- `#define IS_RTC_SECOND(SECOND) ((SECOND) <= 59)`
Макрос проверки попадания значений секунд в допустимый диапазон.
- `#define IS_RTC_MINUTE(MINUTE) ((MINUTE) <= 59)`
Макрос проверки попадания значений минут в допустимый диапазон.
- `#define IS_RTC_HOUR(HOUR) ((HOUR) <= 23)`
Макрос проверки попадания значений часов в допустимый диапазон.
- `#define IS_RTC_WEEKDAY(WEEKDAY)`
Макрос проверки аргументов типа `RTC_Weekday_TypeDef`.
- `#define IS_RTC_DAY(DAY) (((DAY) > 0) && ((DAY) <= 31))`
Макрос проверки попадания значений дней в допустимый диапазон.
- `#define IS_RTC_MONTH(MONTH)`
Макрос проверки аргументов типа `RTC_Month_TypeDef`.
- `#define IS_RTC_YEAR(YEAR) ((YEAR) <= 99)`
Макрос проверки попадания значений лет в допустимый диапазон.
- `#define IS_RTC_FORMAT(FORMAT)`
Макрос проверки аргументов типа `RTC_Format_TypeDef`.

Перечисления

- enum `RTC_Weekday_TypeDef` {
`RTC_Weekday_Monday = ((uint32_t)0x01), RTC_Weekday_Tuesday = ((uint32_t)0x02), R←`
`TC_Weekday_Wednesday = ((uint32_t)0x03), RTC_Weekday_Thursday = ((uint32_t)0x04),`
`RTC_Weekday_Friday = ((uint32_t)0x05), RTC_Weekday_Saturday = ((uint32_t)0x06), R←`
`TC_Weekday_Sunday = ((uint32_t)0x07) }`
Дни недели.
- enum `RTC_Month_TypeDef` {
`RTC_Month_January = ((uint32_t)0x01), RTC_Month_February = ((uint32_t)0x02), RTC←`
`_Month_March = ((uint32_t)0x03), RTC_Month_April = ((uint32_t)0x04),`
`RTC_Month_May = ((uint32_t)0x05), RTC_Month_June = ((uint32_t)0x06), RTC_Month←`
`_July = ((uint32_t)0x07), RTC_Month_August = ((uint32_t)0x08),`
`RTC_Month_September = ((uint32_t)0x09), RTC_Month_October = ((uint32_t)0x10), RT←`
`C_Month_November = ((uint32_t)0x11), RTC_Month_December = ((uint32_t)0x12) }`
Месяцы.
- enum `RTC_Format_TypeDef` { `RTC_Format_BIN, RTC_Format_BCD` }
Формат ввода/вывода времени и даты.

6.66.1 Подробное описание

6.66.2 Макросы

6.66.2.1 #define IS_RTC_FORMAT(FORMAT)

Макроопределение:

```
((FORMAT) == RTC_Format_BIN) || \
((FORMAT) == RTC_Format_BCD))
```

Макрос проверки аргументов типа [RTC_Format_TypeDef](#).

См. определение в файле niietcm4_rtc.h строка 170

6.66.2.2 #define IS_RTC_MONTH(MONTH)

Макроопределение:

```
((MONTH) == RTC_Month_January) || \
((MONTH) == RTC_Month_February) || \
((MONTH) == RTC_Month_March) || \
((MONTH) == RTC_Month_April) || \
((MONTH) == RTC_Month_May) || \
((MONTH) == RTC_Month_June) || \
((MONTH) == RTC_Month_July) || \
((MONTH) == RTC_Month_August) || \
((MONTH) == RTC_Month_September) || \
((MONTH) == RTC_Month_October) || \
((MONTH) == RTC_Month_November) || \
((MONTH) == RTC_Month_December))
```

Макрос проверки аргументов типа [RTC_Month_TypeDef](#).

См. определение в файле niietcm4_rtc.h строка 136

6.66.2.3 #define IS_RTC_WEEKDAY(WEEKDAY)

Макроопределение:

```
((WEEKDAY) == RTC_Weekday_Monday) || \
((WEEKDAY) == RTC_Weekday_Tuesday) || \
((WEEKDAY) == RTC_Weekday_Wednesday) || \
((WEEKDAY) == RTC_Weekday_Thursday) || \
((WEEKDAY) == RTC_Weekday_Friday) || \
((WEEKDAY) == RTC_Weekday_Saturday) || \
((WEEKDAY) == RTC_Weekday_Sunday))
```

Макрос проверки аргументов типа [RTC_Weekday_TypeDef](#).

См. определение в файле niietcm4_rtc.h строка 97

6.66.3 Перечисления

6.66.3.1 enum RTC_Format_TypeDef

Формат ввода/вывода времени и даты.

Элементы перечислений

RTC_Format_BIN Бинарный формат

RTC_Format_BCD Двоично-десятичный формат

См. определение в файле niietcm4_rtc.h строка 159

6.66.3.2 enum RTC_Month_TypeDef

Месяцы.

Элементы перечислений

```
RTC_Month_January  January
RTC_Month_February February
RTC_Month_March    March
RTC_Month_April    April
RTC_Month_May      May
RTC_Month_June     June
RTC_Month_July     July
RTC_Month_August   August
RTC_Month_September September
RTC_Month_October  October
RTC_Month_November November
RTC_Month_December December
```

См. определение в файле niietcm4_rtc.h строка 115

6.66.3.3 enum RTC_Weekday_TypeDef

Дни недели.

Элементы перечислений

```
RTC_Weekday_Monday  Monday
RTC_Weekday_Tuesday Tuesday
RTC_Weekday_Wednesday Wednesday
RTC_Weekday_Thursday Thursday
RTC_Weekday_Friday  Friday
RTC_Weekday_Saturday Saturday
RTC_Weekday_Sunday  Sunday
```

См. определение в файле niietcm4_rtc.h строка 81

6.67 Функции

Функции

- void RTC_GetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)
- void RTC_GetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)
- void RTC_SetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)
- void RTC_SetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)

6.67.1 Подробное описание

6.68 Типы

Макросы

- `#define IS_TIMER_EXT_INPUT(EXT_INPUT)`
Макрос проверки аргументов типа `TIMER_ExtInput_TypeDef`.

Перечисления

- `enum TIMER_ExtInput_TypeDef { TIMER_ExtInput_Disable, TIMER_ExtInput_CountClk, TIMER_ExtInput_CountEn }`
Настройка внешнего тактирования таймера.

6.68.1 Подробное описание

6.68.2 Макросы

6.68.2.1 `#define IS_TIMER_EXT_INPUT(EXT_INPUT)`

Макроопределение:

```
((EXT_INPUT) == TIMER_ExtInput_Disable) || \
((EXT_INPUT) == TIMER_ExtInput_CountClk) || \
((EXT_INPUT) == TIMER_ExtInput_CountEn))
```

Макрос проверки аргументов типа `TIMER_ExtInput_TypeDef`.

См. определение в файле `niietcm4_timer.h` строка 67

Используется в `TIMER_ExtInputConfig()`.

6.68.3 Перечисления

6.68.3.1 `enum TIMER_ExtInput_TypeDef`

Настройка внешнего тактирования таймера.

Элементы перечислений

`TIMER_ExtInput_Disable` Внешнее тактирование не используется.

`TIMER_ExtInput_CountClk` Таймер считает по внешнему тактовому сигналу.

`TIMER_ExtInput_CountEn` Таймер считает по внутреннему тактовому сигналу и только тогда, когда на выводе "1".

См. определение в файле `niietcm4_timer.h` строка 56

6.69 Константы

6.70 Функции

Группы

- [Конфигурация](#)
- [Прерывания](#)

6.70.1 Подробное описание

6.71 Конфигурация

Функции

- void **TIMER_Cmd** (NT_TIMER_TypeDef *TIMERx, [FunctionalState](#) State)
Разрешение работы выбранного таймера.
- void **TIMER_PeriodConfig** (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)
Настройка периода опустошения выбранного таймера.
- void **TIMER_FreqConfig** (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)
Настройка частоты опустошения выбранного таймера.
- void **TIMER_SetReload** (NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)
Установка значения перезагрузки.
- uint32_t **TIMER_GetReload** (NT_TIMER_TypeDef *TIMERx)
Получение текущего значения перезагрузки.
- void **TIMER_SetCounter** (NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)
Установка значения счетчика.
- uint32_t **TIMER_GetCounter** (NT_TIMER_TypeDef *TIMERx)
Получение текущего значения счетчика.
- void **TIMER_ExtInputConfig** (NT_TIMER_TypeDef *TIMERx, [TIMER_ExtInput_TypeDef](#) TIMER_ExtInput)
Выбор режима работы входа внешнего тактирования.

6.71.1 Подробное описание

6.71.2 Функции

6.71.2.1 void TIMER_Cmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 65

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

6.71.2.2 void TIMER_ExtInputConfig (NT_TIMER_TypeDef * TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)

Выбор режима работы входа внешнего тактирования.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

TIMER_Ext↔ Input	Выбор режима работы. Параметр принимает любое значение из TIMER_Ext↔ Input_TypeDef .
---------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 171

Перекрестные ссылки IS_TIMER_ALL_PERIPH, IS_TIMER_EXT_INPUT, TIMER_ExtInput↔_CountClk и TIMER_ExtInput_CountEn.

6.71.2.3 void TIMER_FreqConfig (NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)

Настройка частоты опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемого периода опустошения таймера [TIMER_PeriodConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerFreq	Частота опустошения таймера в Гц.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 102

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.71.2.4 uint32_t TIMER_GetCounter (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_timer.c строка 156

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.71.2.5 uint32_t TIMER_GetReload (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_timer.c строка 129

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.71.2.6 void TIMER_PeriodConfig (NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)

Настройка периода опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемой частоты опустошения таймера [TIMER_FreqConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerPeriod	Период опустошения таймера в мкс.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 84

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.71.2.7 void TIMER_SetCounter (NT_TIMER_TypeDef * TIMERx, uint32_t CounterVal)

Установка значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
CounterVal	Значение счетчика.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 143

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.71.2.8 void TIMER_SetReload (NT_TIMER_TypeDef * TIMERx, uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
ReloadVal	Значение перезагрузки.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 116

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.72 Прерывания

Функции

- void [TIMER_ITCmd](#) (NT_TIMER_TypeDef *TIMERx, [FunctionalState](#) State)
Разрешение работы прерывания выбранного таймера.
- [FlagStatus](#) [TIMER_ITStatus](#) (NT_TIMER_TypeDef *TIMERx)
Чтение статуса прерывания выбранного таймера.
- void [TIMER_ITStatusClear](#) (NT_TIMER_TypeDef *TIMERx)
Очищение статусного бита прерывания выбранного таймера.

6.72.1 Подробное описание

6.72.2 Функции

6.72.2.1 void [TIMER_ITCmd](#) (NT_TIMER_TypeDef * TIMERx, [FunctionalState](#) State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#) и [IS_TIMER_ALL_PERIPH](#).

6.72.2.2 [FlagStatus](#) [TIMER_ITStatus](#) (NT_TIMER_TypeDef * TIMERx)

Чтение статуса прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
------------------------	---

Возвращаемые значения

Status	Статус прерывания.
------------------------	--------------------

См. определение в файле niietcm4_timer.c строка 214

Перекрестные ссылки [IS_TIMER_ALL_PERIPH](#).

6.72.2.3 void [TIMER_ITStatusClear](#) (NT_TIMER_TypeDef * TIMERx)

Очищение статусного бита прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 238

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.73 Типы

Структуры данных

- struct `UART_ModemInit_TypeDef`
Структура инициализации модемного режима.
- struct `UART_Init_TypeDef`
Структура инициализации UART.

Макросы

- #define `IS_UART_INT_DIV(INT_DIV)` (((INT_DIV) > ((uint32_t)0x0)) && ((INT_DIV) < ((uint32_t)0x10000)))
Макрос проверки соответствия величины целой части делителя baudrate UART диапазону.
- #define `IS_UART_FRAC_DIV(FRAC_DIV)` ((FRAC_DIV) < ((uint32_t)0x40))
Макрос проверки соответствия величины дробной части делителя baudrate UART диапазону.
- #define `IS_UART_DATA(DATA)` ((DATA) < ((uint32_t)0x100))
Макрос проверки корректности передаваемых данных.
- #define `IS_UART_FLAG(FLAG)`
Макрос проверки аргументов типа `UART_Flag_TypeDef`.
- #define `IS_UART_ERROR(ERROR)`
Макрос проверки аргументов типа `UART_Error_TypeDef`.
- #define `IS_UART_IT_SOURCE(IT_SOURCE)` (((IT_SOURCE) > ((uint32_t)0x0)) && ((IT_SOURCE) < ((uint32_t)0x800)))
Макрос проверки аргументов типа `UART_ITSource_TypeDef`.
- #define `IS_UART_GET_IT_SOURCE(IT_SOURCE)`
Макрос проверки номера пина при работе с пинами по отдельности.
- #define `IS_UART_DIR(DIR)`
Макрос проверки аргументов типа `UART_Dir_TypeDef`.
- #define `IS_UART_STOP_BIT(STOP_BIT)`
Макрос проверки аргументов типа `UART_StopBit_TypeDef`.
- #define `IS_UART_PARITY_BIT(PARITY_BIT)`
Макрос проверки аргументов типа `UART_ParityBit_TypeDef`.
- #define `IS_UART_DATA_WIDTH(DATA_WIDTH)`
Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.
- #define `IS_UART_FIFO_LEVEL(FIFO_LEVEL)`
Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

Перечисления

- enum `UART_Flag_TypeDef` {
 `UART_Flag_InvCTS`, `UART_Flag_InvDSR`, `UART_Flag_InvDCD`, `UART_Flag_Busy`,
 `UART_Flag_RxFIFOEmpty`, `UART_Flag_TxFIFOFull`, `UART_Flag_RxFIFOFull`, `UART_Flag_TxFIFOEmpty`,
 `UART_Flag_InvRI` }
Перечень флагов.
- enum `UART_Error_TypeDef` { `UART_Error_Frame`, `UART_Error_Parity`, `UART_Error_Break`, `UART_Error_Overflow` }
Перечень ошибок приемника.

- enum `UART_ITSource_TypeDef` {
`UART_ITSource_ChangeRI` = ((uint32_t)0x00000001), `UART_ITSource_ChangeCTS` =
((uint32_t)0x00000002), `UART_ITSource_ChangeDCD` = ((uint32_t)0x00000004), `UART_↵`
`ITSource_ChangeDSR` = ((uint32_t)0x00000008),
`UART_ITSource_RxFIFOLevel` = ((uint32_t)0x00000010), `UART_ITSource_TxFIFOLevel` =
((uint32_t)0x00000020), `UART_ITSource_RecieveTimeout` = ((uint32_t)0x00000040), `UART_↵`
`ITSource_ErrorFrame` = ((uint32_t)0x00000080),
`UART_ITSource_ErrorParity` = ((uint32_t)0x00000100), `UART_ITSource_ErrorBreak` =
((uint32_t)0x00000200), `UART_ITSource_ErrorOverflow` = ((uint32_t)0x00000400) }

Источники прерываний UART.

- enum `UART_Dir_TypeDef` { `UART_Dir_Rx`, `UART_Dir_Tx` }

Направления передачи UART.

- enum `UART_StopBit_TypeDef` { `UART_StopBit_1`, `UART_StopBit_2` }

Выбор режима передачи стопового бита.

- enum `UART_ParityBit_TypeDef` {
`UART_ParityBit_Disable`, `UART_ParityBit_Odd`, `UART_ParityBit_Even`, `UART_Parity_↵`
`Bit_High`,
`UART_ParityBit_Low` }

Выбор режима бита четности.

- enum `UART_DataWidth_TypeDef` { `UART_DataWidth_5`, `UART_DataWidth_6`, `UART_↵`
`DataWidth_7`, `UART_DataWidth_8` }

Количество передаваемых/принимаемых информационных бит.

- enum `UART_FIFOLevel_TypeDef` {
`UART_FIFOLevel_1_8`, `UART_FIFOLevel_1_4`, `UART_FIFOLevel_1_2`, `UART_FIFO_↵`
`Level_3_4`,
`UART_FIFOLevel_7_8` }

Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

6.73.1 Подробное описание

6.73.2 Макросы

6.73.2.1 #define IS_UART_DATA_WIDTH(DATA_WIDTH)

Макроопределение:

```
((DATA_WIDTH) == UART_DataWidth_5) || \
((DATA_WIDTH) == UART_DataWidth_6) || \
((DATA_WIDTH) == UART_DataWidth_7) || \
((DATA_WIDTH) == UART_DataWidth_8))
```

Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.

См. определение в файле `niietcm4_uart.h` строка 236

Используется в `UART_Init()`.

6.73.2.2 #define IS_UART_DIR(DIR)

Макроопределение:

```
((DIR) == UART_Dir_Rx) || \
((DIR) == UART_Dir_Tx))
```

Макрос проверки аргументов типа `UART_Dir_TypeDef`.

См. определение в файле `niietcm4_uart.h` строка 177

Используется в `UART_DMAMCmd()` и `UART_ITFIFOLevelConfig()`.

6.73.2.3 #define IS_UART_ERROR(ERROR)

Макроопределение:

```
((ERROR) == UART_Error_Frame) || \
    ((ERROR) == UART_Error_Parity) || \
    ((ERROR) == UART_Error_Break) || \
    ((ERROR) == UART_Error_Overflow))
```

Макрос проверки аргументов типа `UART_Error_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 117

Используется в `UART_ErrorStatus()`.

6.73.2.4 #define IS_UART_FIFO_LEVEL(FIFO_LEVEL)

Макроопределение:

```
((FIFO_LEVEL) == UART_FIFOLevel_1_8) || \
    ((FIFO_LEVEL) == UART_FIFOLevel_1_4) || \
    ((FIFO_LEVEL) == UART_FIFOLevel_1_2) || \
    ((FIFO_LEVEL) == UART_FIFOLevel_3_4) || \
    ((FIFO_LEVEL) == UART_FIFOLevel_7_8))
```

Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

См. определение в файле `niietcm4_uart.h` строка 259

Используется в `UART_Init()` и `UART_ITFIFOLevelConfig()`.

6.73.2.5 #define IS_UART_FLAG(FLAG)

Макроопределение:

```
((FLAG) == UART_Flag_InvCTS) || \
    ((FLAG) == UART_Flag_InvDSR) || \
    ((FLAG) == UART_Flag_InvDCD) || \
    ((FLAG) == UART_Flag_Busy) || \
    ((FLAG) == UART_Flag_RxFIFOEmpty) || \
    ((FLAG) == UART_Flag_TxFIFOFull) || \
    ((FLAG) == UART_Flag_RxFIFOFull) || \
    ((FLAG) == UART_Flag_TxFIFOEmpty) || \
    ((FLAG) == UART_Flag_InvRI))
```

Макрос проверки аргументов типа `UART_Flag_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 91

Используется в `UART_FlagStatus()`.

6.73.2.6 #define IS_UART_GET_IT_SOURCE(IT_SOURCE)

Макроопределение:

```
((IT_SOURCE) == UART_ITSource_ChangeRI) || \
    ((IT_SOURCE) == \
    UART_ITSource_ChangeCTS) || \
    ((IT_SOURCE) == \
    UART_ITSource_ChangeDCD) || \
    ((IT_SOURCE) == \
    UART_ITSource_ChangeDSR) || \
    ((IT_SOURCE) == \
    UART_ITSource_RxFIFOLevel) || \
    ((IT_SOURCE) == \
    UART_ITSource_TxFIFOLevel) || \
    ((IT_SOURCE) == \
    UART_ITSource_RecieveTimeout) || \
```



```

        ((IT_SOURCE) ==
UART_ITSource_ErrorFrame) || \
        ((IT_SOURCE) ==
UART_ITSource_ErrorParity) || \
        ((IT_SOURCE) ==
UART_ITSource_ErrorBreak) || \
        ((IT_SOURCE) ==
UART_ITSource_ErrorOverflow))

```

Макрос проверки номера пина при работе с пинами по отдельности.

См. определение в файле niietcm4_uart.h строка 151

Используется в UART_ITMaskedStatus() и UART_ITRawStatus().

6.73.2.7 #define IS_UART_PARITY_BIT(PARITY_BIT)

Макроопределение:

```

(((PARITY_BIT) == UART_ParityBit_Disable) || \
 ((PARITY_BIT) == UART_ParityBit_Odd) || \
 ((PARITY_BIT) == UART_ParityBit_Even) || \
 ((PARITY_BIT) == UART_ParityBit_High) || \
 ((PARITY_BIT) == UART_ParityBit_Low))

```

Макрос проверки аргументов типа UART_ParityBit_TypeDef.

См. определение в файле niietcm4_uart.h строка 214

Используется в UART_Init().

6.73.2.8 #define IS_UART_STOP_BIT(STOP_BIT)

Макроопределение:

```

(((STOP_BIT) == UART_StopBit_1) || \
 ((STOP_BIT) == UART_StopBit_2))

```

Макрос проверки аргументов типа UART_StopBit_TypeDef.

См. определение в файле niietcm4_uart.h строка 194

Используется в UART_Init().

6.73.3 Перечисления

6.73.3.1 enum UART_DataWidth_TypeDef

Количество передаваемых/принимаемых информационных бит.

Элементы перечислений

UART_DataWidth_5 Длина информационного слова 5 бит.

UART_DataWidth_6 Длина информационного слова 6 бит.

UART_DataWidth_7 Длина информационного слова 7 бит.

UART_DataWidth_8 Длина информационного слова 8 бит.

См. определение в файле niietcm4_uart.h строка 224

6.73.3.2 enum UART_Dir_Typedef

Направления передачи UART.

Элементы перечислений

UART_Dir_Rx Передача.

UART_Dir_Tx Прием.

См. определение в файле niietcm4_uart.h строка 167

6.73.3.3 enum UART_Error_Typedef

Перечень ошибок приемника.

Элементы перечислений

UART_Error_Frame Флаг ошибки в структуре кадра.

UART_Error_Parity Флаг ошибки контроля четности.

UART_Error_Break Флаг разрыва линии.

UART_Error_Overflow Флаг переполнения буфера приемника.

См. определение в файле niietcm4_uart.h строка 105

6.73.3.4 enum UART_FIFOLevel_TypeDef

Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

Элементы перечислений

UART_FIFOLevel_1_8 Заполнение FIFO на 1/8.

UART_FIFOLevel_1_4 Заполнение FIFO на 1/4.

UART_FIFOLevel_1_2 Заполнение FIFO на 1/2.

UART_FIFOLevel_3_4 Заполнение FIFO на 3/4.

UART_FIFOLevel_7_8 Заполнение FIFO на 7/8.

См. определение в файле niietcm4_uart.h строка 246

6.73.3.5 enum UART_Flag_Typedef

Перечень флагов.

Элементы перечислений

UART_Flag_InvCTS Флаг инверсии сигнала на линии UART_CTS.

UART_Flag_InvDSR Флаг инверсии сигнала на линии UART_DSR.

UART_Flag_InvDCD Флаг инверсии сигнала на линии UART_DSR.

UART_Flag_Busy Флаг занятости блока UART.

UART_Flag_RxFIFOEmpty Флаг пустоты буфера приемника.

UART_Flag_TxFIFOFull Флаг заполнения буфера передатчика.

UART_Flag_RxFIFOFull Флаг заполнения буфера приемника.

UART_Flag_TxFIFOEmpty Флаг пустоты буфера передатчика.

UART_Flag_InvRI Флаг инверсии сигнала на линии UART_RI.

См. определение в файле niietcm4_uart.h строка 74

6.73.3.6 enum UART_ITSource_TypeDef

Источники прерываний UART.

Элементы перечислений

UART_ITSource_ChangeRI Изменение состояния линии UART_RI
UART_ITSource_ChangeCTS Изменение состояния линии UART_CTS
UART_ITSource_ChangeDCD Изменение состояния линии UART_DCD
UART_ITSource_ChangeDSR Изменение состояния линии UART_DSR
UART_ITSource_RxFIFOLevel Порог переполнения буфера приемника
UART_ITSource_TxFIFOLevel Порог опустошения буфера передатчика
UART_ITSource_RecieveTimeout Таймаут приема данных
UART_ITSource_ErrorFrame Ошибка в структуре кадра
UART_ITSource_ErrorParity Ошибка контроля четности
UART_ITSource_ErrorBreak Разрыв линии
UART_ITSource_ErrorOverflow Переполнение буфера приемника

См. определение в файле niietcm4_uart.h строка 126

6.73.3.7 enum UART_ParityBit_TypeDef

Выбор режима бита четности.

Элементы перечислений

UART_ParityBit_Disable Не передается, не проверяется.
UART_ParityBit_Odd Проверка нечетности данных.
UART_ParityBit_Even Проверка четности данных.
UART_ParityBit_High Бит четности постоянно равен единице.
UART_ParityBit_Low Бит четности постоянно равен нулю.

См. определение в файле niietcm4_uart.h строка 201

6.73.3.8 enum UART_StopBit_TypeDef

Выбор режима передачи стопового бита.

Элементы перечислений

UART_StopBit_1 Один стоповый бит.
UART_StopBit_2 Два стоповых бита.

См. определение в файле niietcm4_uart.h строка 184

6.74 Константы

6.75 Функции

Группы

- [Инициализация и деинициализация](#)
- [Прием и передача](#)
- [Режим модема](#)
- [Прерывания](#)
- [Настройка DMA](#)

Функции

- void [UART_Cmd](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Разрешение работы выбранного UART.
- void [UART_BaudRateDivConfig](#) (NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t ←
t FracDiv)
Ручная настройка делителя для реализации необходимой скорости передачи.
- void [UART_Break](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Включение разрыва линии.

6.75.1 Подробное описание

6.75.2 Функции

6.75.2.1 void [UART_BaudRateDivConfig](#) (NT_UART_TypeDef * UARTx, uint32_t IntDiv,
uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

6.75.2.2 void [UART_Break](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 106

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.75.2.3 `void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)`

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 69

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.76 Инициализация и деинициализация

Функции

- void [UART_DeInit](#) (NT_UART_TypeDef *UARTx)
Устанавливает все регистры UART значениями по умолчанию.
- [OperationStatus](#) [UART_Init](#) (NT_UART_TypeDef *UARTx, [UART_Init_TypeDef](#) *UART_InitStruct)
Инициализирует UARTx согласно параметрам структуры UART_InitStruct.
- void [UART_StructInit](#) ([UART_Init_TypeDef](#) *UART_InitStruct)
Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

6.76.1 Подробное описание

6.76.2 Функции

6.76.2.1 void UART_DeInit (NT_UART_TypeDef * UARTx)

Устанавливает все регистры UART значениями по умолчанию.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 120

Перекрестные ссылки IS_UART_ALL_PERIPH, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_UART2, RCC_PeriphRst_UART3 и RCC_PeriphRstCmd().

6.76.2.2 [OperationStatus](#) [UART_Init](#) (NT_UART_TypeDef * UARTx, [UART_Init_TypeDef](#) * UART_InitStruct)

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле niietcm4_uart.c строка 159

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DATA_WIDTH, IS_UART_FIFO_LEVEL, IS_UART_PARITY_BIT, IS_UART_STOP_BIT, [UART_Init_TypeDef::UART_BaudRate](#), [UART_Init_TypeDef::UART_ClkFreq](#), [UART_Init_TypeDef::UART_DataWidth](#), [UART_Init_TypeDef::UART_FIFOEn](#), [UART_Init_TypeDef::UART_FIFOLevelRx](#), [UART_Init_TypeDef::UART_FIFOLevelTx](#), [UART_Init_TypeDef::UART_ParityBit](#), [UART_ParityBit_Even](#), [UART_ParityBit_High](#), [UART_ParityBit_Low](#), [UART_ParityBit_Odd](#), [UART_Init_TypeDef::UART_RxEn](#), [UART_Init_TypeDef::UART_StopBit](#) и [UART_Init_TypeDef::UART_TxEn](#).

6.76.2.3 void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

Аргументы

UART_Init↵ Struct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
----------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_uart.c` строка 229

Перекрестные ссылки EXT_OSC_VALUE, UART_Init_TypeDef::UART_BaudRate, UART_Init↵
__TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_DataWidth_8, U↵
ART_Init_TypeDef::UART_FIFOEn, UART_FIFOLevel_1_2, UART_Init_TypeDef::UART_F↵
IFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit,
UART_ParityBit_Disable, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_↵
StopBit, UART_StopBit_1 и UART_Init_TypeDef::UART_TxEn.

6.77 Прием и передача

Функции

- void [UART_SendData](#) (NT_UART_TypeDef *UARTx, uint32_t Data)
Передача слова данных.
- uint32_t [UART_RecieveData](#) (NT_UART_TypeDef *UARTx)
Прием слова данных.
- FlagStatus [UART_FlagStatus](#) (NT_UART_TypeDef *UARTx, [UART_Flag_Typedef](#) UART←_Flag)
Запрос состояния выбранного флага.
- FlagStatus [UART_ErrorStatus](#) (NT_UART_TypeDef *UARTx, [UART_Error_Typedef](#) UART←_Error)
Запрос состояния выбранного флага ошибки.
- void [UART_ErrorStatusClear](#) (NT_UART_TypeDef *UARTx)
Очистка флагов ошибки.

6.77.1 Подробное описание

6.77.2 Функции

6.77.2.1 FlagStatus [UART_ErrorStatus](#) (NT_UART_TypeDef * UARTx, [UART_Error_Typedef](#) UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error←_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

6.77.2.2 void [UART_ErrorStatusClear](#) (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.77.2.3 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_TypeDef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

6.77.2.4 uint32_t UART_RcieveData (NT_UART_TypeDef * UARTx)

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.77.2.5 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

6.78 Режим модема

Функции

- void [UART_ModemConfig](#) (NT_UART_TypeDef *UARTx, [UART_ModemInit_TypeDef](#) *UART_ModemInitStruct)
Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.
- void [UART_ModemStructInit](#) ([UART_ModemInit_TypeDef](#) *UART_ModemInitStruct)
Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

6.78.1 Подробное описание

6.78.2 Функции

6.78.2.1 void [UART_ModemConfig](#) (NT_UART_TypeDef * UARTx, [UART_ModemInit_TypeDef](#) * [UART_ModemInitStruct](#))

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.

Аргументы

UART_ModemInitStruct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
--------------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_UART_ALL_PERIPH](#), [UART_ModemInit_TypeDef::UART_CTSEn](#), [UART_ModemInit_TypeDef::UART_InvDTR](#), [UART_ModemInit_TypeDef::UART_InvRTS](#) и [UART_ModemInit_TypeDef::UART_RTSEn](#).

6.78.2.2 void [UART_ModemStructInit](#) ([UART_ModemInit_TypeDef](#) * [UART_ModemInitStruct](#))

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_ModemInitStruct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
--------------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки [UART_ModemInit_TypeDef::UART_CTSEn](#), [UART_ModemInit_TypeDef::UART_InvDTR](#), [UART_ModemInit_TypeDef::UART_InvRTS](#) и [UART_ModemInit_TypeDef::UART_RTSEn](#).

6.79 Прерывания

Функции

- void [UART_ITFIFOLevelConfig](#) (NT_UART_TypeDef *UARTx, [UART_Dir_Typedef](#) UART_Dir, [UART_FIFOLevel_TypeDef](#) UART_FIFOLevel)
Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.
- void [UART_ITCmd](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource, [FunctionalState](#) State)
Маскирование выбранных прерываний.
- [FlagStatus](#) [UART_ITRawStatus](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource)
Запрос немаскированного состояния прерывания.
- [FlagStatus](#) [UART_ITMaskedStatus](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource)
Запрос маскированного состояния прерывания.
- void [UART_ITStatusClear](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource)
Сброс флагов состояния выбранных прерываний.

6.79.1 Подробное описание

6.79.2 Функции

6.79.2.1 void [UART_ITCmd](#) (NT_UART_TypeDef * UARTx, [UART_ITSource_Typedef](#) UART_ITSource, [FunctionalState](#) State)

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_uart.c` строка 410

Перекрестные ссылки `IS_UART_ALL_PERIPH` и `IS_UART_IT_SOURCE`.

6.79.2.2 void [UART_ITFIFOLevelConfig](#) (NT_UART_TypeDef * UARTx, [UART_Dir_Typedef](#) UART_Dir, [UART_FIFOLevel_TypeDef](#) UART_FIFOLevel)

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

UART_Dir	Выбор между буфером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFO↔ OLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_↔ TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и U↔
ART_Dir_Rx.

6.79.2.3 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

6.79.2.4 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

6.79.2.5 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

6.80 Настройка DMA

Функции

- void [UART_DMABlkOnErrCmd](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.
- void [UART_DMAMCmd](#) (NT_UART_TypeDef *UARTx, [UART_Dir_Typedef](#) UART_Dir, [FunctionalState](#) State)
Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

6.80.1 Подробное описание

6.80.2 Функции

6.80.2.1 void [UART_DMABlkOnErrCmd](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.80.2.2 void [UART_DMAMCmd](#) (NT_UART_TypeDef * UARTx, [UART_Dir_Typedef](#) UART_Dir, [FunctionalState](#) State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DIR и UART_Dir_Rx.

6.81 Константы

Группы

- [Основная область флеш](#)
- [Информационная область флеш](#)

Макросы

- `#define USERFLASH_OPERATION_TIMEOUT ((uint32_t)10000000)`
Время ожидания выполнения операции с флеш.
- `#define USERFLASH_MAGIC_KEY ((uint32_t)0xA4420000)`
Ключ для проведения операций с контроллером пользовательской флеш.

6.81.1 Подробное описание

6.82 Основная область флеш

Макросы

- `#define USERFLASH_PAGE_SIZE_BYTES ((uint32_t)256)`
- `#define USERFLASH_PAGE_TOTAL ((uint32_t)256)`
- `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`
- `#define IS_USERFLASH_PAGE_NUM(PAGE_NUM) (PAGE_NUM < USERFLASH_PAGE_TOTAL)`

Макрос проверки номера страницы основной области пользовательской флеш на попадание в допустимый диапазон.

6.82.1 Подробное описание

6.82.2 Макросы

6.82.2.1 `#define USERFLASH_PAGE_SIZE_BYTES ((uint32_t)256)`

Размер страницы в байтах.

См. определение в файле `niietcm4_userflash.h` строка 68

Используется в `USERFLASH_Info_PageErase()` и `USERFLASH_PageErase()`.

6.82.2.2 `#define USERFLASH_PAGE_TOTAL ((uint32_t)256)`

Общее количество страниц.

См. определение в файле `niietcm4_userflash.h` строка 69

6.82.2.3 `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`

Общий размер основной области.

См. определение в файле `niietcm4_userflash.h` строка 70

6.83 Информационная область флеш

Макросы

- `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES`
- `#define USERFLASH_INFO_PAGE_TOTAL ((uint32_t)2)`
- `#define USERFLASH_INFO_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`
- `#define IS_USERFLASH_INFO_PAGE_NUM(PAGE_NUM) (PAGE_NUM < USERFLASH_INFO_PAGE_TOTAL)`

Макрос проверки номера страницы информационной области пользовательской флеш на попадание в допустимый диапазон.

6.83.1 Подробное описание

6.83.2 Макросы

6.83.2.1 `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES`

Размер страницы в байтах.

См. определение в файле `niietcm4_userflash.h` строка 86

6.83.2.2 `#define USERFLASH_INFO_PAGE_TOTAL ((uint32_t)2)`

Общее количество страниц.

См. определение в файле `niietcm4_userflash.h` строка 87

6.83.2.3 `#define USERFLASH_INFO_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`

Общий размер информационной области.

См. определение в файле `niietcm4_userflash.h` строка 88

6.84 Типы

Макросы

- `#define IS_USERFLASH_STATUS(STATUS)`
Макрос проверки аргументов типа `USERFLASH_Status_TypeDef`.

Перечисления

- `enum USERFLASH_Status_TypeDef { USERFLASH_Status_None = ((uint32_t)0), USERFLASH_Status_Complete = ((uint32_t)1), USERFLASH_Status_Error = ((uint32_t)3) }`
Статус работы контроллера пользовательской флеш-памяти.

6.84.1 Подробное описание

6.84.2 Макросы

6.84.2.1 `#define IS_USERFLASH_STATUS(STATUS)`

Макроопределение:

```
((STATUS) == USERFLASH_Status_None) || \
((STATUS) == USERFLASH_Status_Complete) || \
((STATUS) == USERFLASH_Status_Error))
```

Макрос проверки аргументов типа `USERFLASH_Status_TypeDef`.

См. определение в файле `niietcm4_userflash.h` строка 123

6.84.3 Перечисления

6.84.3.1 `enum USERFLASH_Status_TypeDef`

Статус работы контроллера пользовательской флеш-памяти.

Элементы перечислений

`USERFLASH_Status_None` Операция выполняется или отсутствует.

`USERFLASH_Status_Complete` Операция успешно завершена.

`USERFLASH_Status_Error` Операция завершена с ошибкой.

См. определение в файле `niietcm4_userflash.h` строка 112

6.85 Функции

Группы

- [Основная область флеш](#)
- [Информационная область флеш](#)

Функции

- void [USERFLASH_Init](#) (uint32_t SysClkFreq)
Инициализирует тайминги доступа для контроллера пользовательской флеш.
- [USERFLASH_Status_TypeDef USERFLASH_OperationStatus](#) ()
Статус работы контроллера пользовательской флеш.
- void [USERFLASH_OperationStatusClear](#) ()
Очищает статус работы контроллера пользовательской флеш.
- void [USERFLASH_ITCmd](#) (FunctionalState State)
Включение прерывания по завершению чтения/записи/стирания.

6.85.1 Подробное описание

6.85.2 Функции

6.85.2.1 void [USERFLASH_Init](#) (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера пользовательской флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
----------------------------	---------------------------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 66

6.85.2.2 void [USERFLASH_ITCmd](#) (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-----------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.85.2.3 [USERFLASH_Status_TypeDef USERFLASH_OperationStatus](#) ()

Статус работы контроллера пользовательской флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из USERFLASH↵ _Status_TypeDef .
--------	---

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

6.85.2.4 void USERFLASH_OperationStatusClear ()

Очищает статус работы контроллера пользовательской флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 93

Используется в USERFLASH_Info_Read() и USERFLASH_Read().

6.86 Основная область флеш

Функции

- uint32_t [USERFLASH_Read](#) (uint32_t Address)
Чтение байта из основной области пользовательской флеш.
- void [USERFLASH_Write](#) (uint32_t Address, uint32_t Data)
Запись байта в основную область пользовательской флеш по указанному адресу.
- void [USERFLASH_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы основной области пользовательской флеш.
- void [USERFLASH_FullErase](#) ()
Полная очистка основной области пользовательской флеш.

6.86.1 Подробное описание

6.86.2 Функции

6.86.2.1 void [USERFLASH_FullErase](#) ()

Полная очистка основной области пользовательской флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 103

Перекрестные ссылки [USERFLASH_MAGIC_KEY](#), [USERFLASH_OperationStatus\(\)](#) и [USERFLASH_Status_None](#).

6.86.2.2 void [USERFLASH_PageErase](#) (uint32_t PageNum)

Стирание указанной страницы основной области пользовательской флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 161

Перекрестные ссылки [IS_USERFLASH_PAGE_NUM](#), [USERFLASH_MAGIC_KEY](#) и [USERFLASH_PAGE_SIZE_BYTES](#).

6.86.2.3 uint32_t [USERFLASH_Read](#) (uint32_t Address)

Чтение байта из основной области пользовательской флеш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 116

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

6.86.2.4 void USERFLASH_Write (uint32_t Address, uint32_t Data)

Запись байта в основную область пользовательской флеш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 148

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.87 Информационная область флеш

Функции

- `uint32_t USERFLASH_Info_Read (uint32_t Address)`
Чтение байта из информационной области пользовательской флеш.
- `void USERFLASH_Info_Write (uint32_t Address, uint32_t Data)`
Запись байта в информационную область пользовательской флеш по указанному адресу.
- `void USERFLASH_Info_PageErase (uint32_t PageNum)`
Стирание указанной страницы информационной области пользовательской флеш.

6.87.1 Подробное описание

6.87.2 Функции

6.87.2.1 `void USERFLASH_Info_PageErase (uint32_t PageNum)`

Стирание указанной страницы информационной области пользовательской флеш.

Аргументы

<code>PageNum</code>	Номер страницы.
----------------------	-----------------

Возвращаемые значения

Нет

См. определение в файле `niietcm4_userflash.c` строка 219

Перекрестные ссылки `IS_USERFLASH_INFO_PAGE_NUM`, `USERFLASH_MAGIC_KEY` и `USERFLASH_PAGE_SIZE_BYTES`.

6.87.2.2 `uint32_t USERFLASH_Info_Read (uint32_t Address)`

Чтение байта из информационной области пользовательской флеш.

Аргументы

<code>Address</code>	Адрес чтения.
----------------------	---------------

Возвращаемые значения

<code>Data</code>	Байт данных.
-------------------	--------------

См. определение в файле `niietcm4_userflash.c` строка 175

Перекрестные ссылки `USERFLASH_MAGIC_KEY`, `USERFLASH_OPERATION_TIMEOUT`, `USERFLASH_OperationStatus()`, `USERFLASH_OperationStatusClear()` и `USERFLASH_Status_None`.

6.87.2.3 `void USERFLASH_Info_Write (uint32_t Address, uint32_t Data)`

Запись байта в информационную область пользовательской флеш по указанному адресу.

Аргументы

<code>Address</code>	Адрес записи.
----------------------	---------------

Data	Байт данных.
------	--------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 206

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.88 Типы

Макросы

- `#define IS_WATCHDOG_RELOAD(RELOAD) ((RELOAD) > ((uint32_t)0x0))`
Макрос проверки соответствия величины значения перезагрузки диапазону.

6.88.1 Подробное описание

6.89 Константы

6.90 Функции

Группы

- [Конфигурация](#)
- [Прерывания](#)

6.90.1 Подробное описание

6.91 Конфигурация

Функции

- void [WATCHDOG_Cmd](#) ([FunctionalState](#) State)
Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.
- void [WATCHDOG_SetReload](#) (uint32_t ReloadVal)
Установка значения перезагрузки.
- uint32_t [WATCHDOG_GetReload](#) ()
Получение текущего значения перезагрузки.
- uint32_t [WATCHDOG_GetCounter](#) ()
Получение текущего значения счетчика.
- void [WATCHDOG_RstCmd](#) ([FunctionalState](#) State)
Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.
- void [WATCHDOG_LockCmd](#) ([FunctionalState](#) State)
Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.

6.91.1 Подробное описание

6.91.2 Функции

6.91.2.1 void [WATCHDOG_Cmd](#) ([FunctionalState](#) State)

Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_watchdog.c` строка 69

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.91.2.2 uint32_t [WATCHDOG_GetCounter](#) ()

Получение текущего значения счетчика.

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле `niietcm4_watchdog.c` строка 105

6.91.2.3 uint32_t [WATCHDOG_GetReload](#) ()

Получение текущего значения перезагрузки.

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_watchdog.c строка 95

6.91.2.4 void WATCHDOG_LockCmd (FunctionalState State)

Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 134

Перекрестные ссылки IS_FUNCTIONAL_STATE, WATCHDOG_Lock_Value и WATCHDOG_Unlock_Value.

6.91.2.5 void WATCHDOG_RstCmd (FunctionalState State)

Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 119

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.91.2.6 void WATCHDOG_SetReload (uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

ReloadVal	Значение перезагрузки. Параметр принимает любое значение из диапазона 0x1 - 0xFFFFFFFF.
-----------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 83

Перекрестные ссылки IS_WATCHDOG_RELOAD.

6.92 Прерывания

Функции

- [FlagStatus WATCHDOG_ITRawStatus \(\)](#)
Чтение немаскированного флага прерывания сторожевого таймера.
- [FlagStatus WATCHDOG_ITMaskedStatus \(\)](#)
Чтение маскированного флага прерывания сторожевого таймера.
- [void WATCHDOG_ITStatusClear \(\)](#)
Очищение статусного бита прерывания сторожевого таймера.

6.92.1 Подробное описание

6.92.2 Функции

6.92.2.1 FlagStatus WATCHDOG_ITMaskedStatus ()

Чтение маскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 174

6.92.2.2 FlagStatus WATCHDOG_ITRawStatus ()

Чтение немаскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 153

6.92.2.3 void WATCHDOG_ITStatusClear ()

Очищение статусного бита прерывания сторожевого таймера.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_watchdog.c строка 195

6.93 ADC

Драйвер для модулей АЦП, связанных с ними секвенсоров, а также цифровых компараторов.

Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.93.1 Подробное описание

Драйвер для модулей АЦП, связанных с ними секвенсоров, а также цифровых компараторов.

Внимание

Драйвер позволяет управлять только внутренними настройками модулей АЦП. Системное тактирование необходимо настраивать отдельно с помощью модуля [RCC](#) :

- [Тактирование ADC](#).

Для корректного функционирования АЦП, перед началом его работы необходимо записать записать во все поля выбора канала, подключаемого к цифровому компаратору, запрещенное значение 0x18-0x1F. С точки зрения драйвера, это проще всего сделать, вызвав функцию [ADC_DC_DeInit\(\)](#) для каждого компаратора.

Драйвер по умолчанию устанавливает отличную от нуля задержку перезапуска модулей АЦП секвенсорами. Минимальная рекомендуемая величина задержки равна 2. Если используется один секвенсор, либо несколько, запускающиеся только синхронно - данную задержку можно убрать. Если используется больше одного секвенсора и начинают измерения они асинхронно, то каждый из них должен иметь задержку как минимум в 2 такта между перезапусками.

Настоятельно рекомендуется использовать только один секвенсор, либо несколько секвенсоров, но работающих только синхронно (один источник запуска, одинаковые задержки перезапуска).

Общий вид процесса инициализации:

- для всех компараторов вызываем функцию деинициализации [ADC_DC_DeInit\(\)](#), чтобы записать в регистр выбора канала, подключаемого к компаратору, запрещенное значение;
- (опционально) инициализируем цифровые компараторы ([Инициализация](#) >> [Цифровые компараторы](#));
- инициализируем необходимые модули АЦП ([Инициализация](#) >> [Модули АЦП](#));
- инициализируем нужное количество секвенсоров ([Инициализация](#) >> [Секвенсоры](#));
- (опционально) настраиваем и включаем функцию работы с DMA для секвенсоров с помощью [Конфигурация секвенсоров для DMA](#);
- (опционально) настраиваем и включаем прерывания цифровых компараторов и секвенсоров через [Конфигурация прерываний](#);
- включаем секвенсоры;
- АЦП готов выполнять измерения по запросам.

Более подробно инициализация и использование АЦП показаны в приложенных к драйверу примерах.

6.94 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.94.1 Подробное описание

6.95 Приватные константы

Группы

- [Начальные значения регистров](#)

6.95.1 Подробное описание

6.96 Начальные значения регистров

6.97 Приватные функции

Функции

- void `ADC_Cmd` (`ADC_Module_TypeDef` `ADC_Module`, `FunctionalState` `State`)
Включение модуля АЦП.
- void `ADC_DeInit` (`ADC_Module_TypeDef` `ADC_Module`)
Устанавливает все регистры модуля АЦП значениями по умолчанию.
- void `ADC_Init` (`ADC_Module_TypeDef` `ADC_Module`, `ADC_Init_TypeDef` *`ADC_InitStruct`)
Инициализирует выбранный модуль АЦП согласно параметрам структуры `ADC_InitStruct`.
- void `ADC_StructInit` (`ADC_Init_TypeDef` *`ADC_InitStruct`)
Заполнение каждого члена структуры `ADC_InitStruct` значениями по умолчанию.
- void `ADC_DC_DeInit` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`)
Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.
- void `ADC_DC_Init` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `ADC_DC_Init_TypeDef` *`ADC_DC_InitStruct`)
Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры `ADC_DC_InitStruct`.
- void `ADC_DC_StructInit` (`ADC_DC_Init_TypeDef` *`ADC_DC_InitStruct`)
Заполнение каждого члена структуры `ADC_DC_InitStruct` значениями по умолчанию.
- void `ADC_SEQ_DeInit` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`)
Устанавливает все регистры выбранного секвенсора значениями по умолчанию.
- void `ADC_SEQ_Init` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`, `ADC_SEQ_Init_TypeDef` *`ADC_SEQ_InitStruct`)
Инициализирует выбранный секвенсор согласно параметрам структуры `ADC_SEQ_InitStruct`.
- void `ADC_SEQ_StructInit` (`ADC_SEQ_Init_TypeDef` *`ADC_SEQ_InitStruct`)
Заполнение каждого члена структуры `ADC_SEQ_InitStruct` значениями по умолчанию.
- void `ADC_SEQ_DMAConfig` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`, `ADC_SEQ_FIFOLevel_TypeDef` `ADC_SEQ_FIFOLevel`)
Конфигурирует выбранный секвенсор для работы с DMA.
- void `ADC_SEQ_DMACmd` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`, `FunctionalState` `State`)
Включает для выбранного секвенсора генерирование запросов DMA.
- `FlagStatus` `ADC_SEQ_DMAErrorStatus` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`)
Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.
- void `ADC_SEQ_DMAErrorStatusClear` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`)
Сброс статуса ошибки DMA.
- void `ADC_DC_ITGenCmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)
Разрешает компаратору генерировать сигнал прерывания.
- void `ADC_DC_ITMaskCmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)
Маскирование сигнала прерывания цифрового компаратора.
- void `ADC_DC_ITCmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)
Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: `ADC_DC_ITGenCmd` и `ADC_DC_ITMaskCmd`.
- void `ADC_DC_ITConfig` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `ADC_DC_Mode_TypeDef` `ADC_DC_Mode`, `ADC_DC_Condition_TypeDef` `ADC_DC_Condition`)
Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.

- [FlagStatus ADC_DC_ITRawStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Проверка флагов немаскированных прерываний.
- [FlagStatus ADC_DC_ITMaskedStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Проверка флагов маскированных прерываний.
- [void ADC_DC_ITStatusClear](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.
- [void ADC_SEQ_ITCmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)
Включение прерывания секвенсора.
- [void ADC_SEQ_ITConfig](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [uint32_t](#) ADC_SEQ_ITRate, [FunctionalState](#) ADC_SEQ_ITCountSEQRst)
Настройка вызова прерывания секвенсора.
- [uint32_t ADC_SEQ_GetITCount](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.
- [void ADC_SEQ_ITCountRst](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Сброс счетчика прерываний секвенсора.
- [FlagStatus ADC_SEQ_ITRawStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка флагов немаскированных прерываний.
- [FlagStatus ADC_SEQ_ITMaskedStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка флагов маскированных прерываний.
- [void ADC_SEQ_ITStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.
- [void ADC_SEQ_Cmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)
Включение секвенсора.
- [void ADC_SEQ_SWReq](#) ()
Программный запуск измерений всех разрешенных секвенсоров.
- [uint32_t ADC_SEQ_GetFIFOData](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Получение результата измерений из буфера секвенсора.
- [uint32_t ADC_SEQ_GetConversionCount](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.
- [uint32_t ADC_SEQ_GetFIFOLoad](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Получение количества измерений, сохраненных в буфере секвенсора.
- [FlagStatus ADC_SEQ_FIFOFullStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.
- [void ADC_SEQ_FIFOFullStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Сброс флага заполнения буфера секвенсора.
- [FlagStatus ADC_SEQ_FIFOEmptyStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.
- [void ADC_SEQ_FIFOEmptyStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
Сброс флага пустоты буфера секвенсора.
- [void ADC_DC_Cmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
Включение выходного триггера цифрового компаратора.
- [uint32_t ADC_DC_GetLastData](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.
- [FlagStatus ADC_DC_TrigStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Проверка состояния выходного триггера компаратора.
- [void ADC_DC_TrigStatusClear](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Сброс выходного триггера цифрового компаратора.

6.97.1 Подробное описание

6.97.2 Функции

6.97.2.1 void ADC_Cmd (ADC_Module_TypeDef ADC_Module, FunctionalState State)

Включение модуля АЦП.

Аргументы

ADC_Module	Выбор АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 108

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

6.97.2.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1017

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

6.97.2.3 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 300

Перекрестные ссылки IS_ADC_DC_MODULE.

6.97.2.4 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 1033

Перекрестные ссылки IS_ADC_DC_MODULE.

```
6.97.2.5 void ADC_DC_Init ( ADC_DC_Module_TypeDef ADC_DC_Module,
                          ADC_DC_Init_TypeDef * ADC_DC_InitStruct )
```

Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры ADC_DC_InitStruct.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 319

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Init_TypeDef::ADC_DC_Mode, ADC_DC_Init_TypeDef::ADC_DC_ThresholdHigh, ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow, IS_ADC_DC, IS_ADC_DC_CHANNEL, IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_THRESHOLD.

```
6.97.2.6 void ADC_DC_ITCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                          FunctionalState State )
```

Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: [ADC_DC_ITGenCmd](#) и [ADC_DC_ITMaskCmd](#).

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 589

Перекрестные ссылки ADC_DC_ITGenCmd(), ADC_DC_ITMaskCmd(), IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

```
6.97.2.7 void ADC_DC_ITConfig ( ADC_DC_Module_TypeDef ADC_DC_Module,
                              ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef
                              ADC_DC_Condition )
```

Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
ADC_DC_↔ Mode	Режим срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Mode_TypeDef .
ADC_DC_↔ Condition	Условие срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Condition_TypeDef .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 613

Перекрестные ссылки IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_MODULE.

```
6.97.2.8 void ADC_DC_ITGenCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                               FunctionalState State )
```

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 546

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

```
6.97.2.9 void ADC_DC_ITMaskCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                                FunctionalState State )
```

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 563

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

```
6.97.2.10 FlagStatus ADC_DC_ITMaskedStatus ( ADC_DC_Module_TypeDef ADC_DC_Module
                                              )
```

Проверка флагов маскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 655

Перекрестные ссылки IS_ADC_DC_MODULE.

6.97.2.11 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 630

Перекрестные ссылки IS_ADC_DC_MODULE.

6.97.2.12 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 681

Перекрестные ссылки IS_ADC_DC_MODULE.

6.97.2.13 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 342

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Channel_None, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Condition_Low, ADC_DC_Init_↔

_TypeDef::ADC_DC_Mode, ADC_DC_Mode_Single, ADC_DC_Init_TypeDef::ADC_DC_↔ThresholdHigh и ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow.

6.97.2.14 FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние триггера.
------------	-----------------------------

См. определение в файле niietcm4_adc.c строка 1051

Перекрестные ссылки IS_ADC_DC_MODULE.

6.97.2.15 void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Сброс выходного триггера цифрового компаратора.

Внимание

Одновременно со сбросом триггеров 0 и 1 компаратора сбрасываются триггеры 10 и 11 компаратора соответственно. То же самое справедливо и для обратного случая. Это происходит аппаратно и программными методами не обходится.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1079

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

6.97.2.16 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_↔TypeDef .
------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 123

Перекрестные ссылки ADC_Module_0, ADC_Module_1, ADC_Module_10, ADC_Module_11, A↔DC_Module_2, ADC_Module_3, ADC_Module_4, ADC_Module_5, ADC_Module_6, ADC_↔Module_7, ADC_Module_8, ADC_Module_9 и IS_ADC_MODULE.

```
6.97.2.17 void ADC_Init ( ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *  
ADC_InitStruct )
```

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
ADC_InitStruct	Указатель на структуру типа ADC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 199

Перекрестные ссылки [ADC_Init_TypeDef::ADC_Average](#), [ADC_Init_TypeDef::ADC_Measure_A](#), [ADC_Init_TypeDef::ADC_Measure_B](#), [ADC_Init_TypeDef::ADC_Mode](#), [ADC_Module_0](#), [ADC_Module_1](#), [ADC_Module_10](#), [ADC_Module_11](#), [ADC_Module_2](#), [ADC_Module_3](#), [ADC_Module_4](#), [ADC_Module_5](#), [ADC_Module_6](#), [ADC_Module_7](#), [ADC_Module_8](#), [ADC_Module_9](#), [ADC_Init_TypeDef::ADC_Phase](#), [ADC_Init_TypeDef::ADC_Resolution](#), [IS_ADC_AVERAGE](#), [IS_ADC_MEASURE](#), [IS_ADC_MODE](#), [IS_ADC_MODULE](#), [IS_ADC_PHASE](#) и [IS_ADC_RESOLUTION](#).

```
6.97.2.18 void ADC_SEQ_Cmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                          FunctionalState State )
```

Включение секвенсора.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 848

Перекрестные ссылки [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

```
6.97.2.19 void ADC_SEQ_DeInit ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module )
```

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

ADC_SEQ_Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_SEQ_TypeDef .
----------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 358

Перекрестные ссылки [ADC_SEQ_Module_0](#), [ADC_SEQ_Module_1](#), [ADC_SEQ_Module_2](#), [ADC_SEQ_Module_3](#), [ADC_SEQ_Module_4](#), [ADC_SEQ_Module_5](#), [ADC_SEQ_Module_6](#), [ADC_SEQ_Module_7](#) и [IS_ADC_SEQ_MODULE](#).

```
6.97.2.20 void ADC_SEQ_DMAMCmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                              FunctionalState State )
```

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 489

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

```
6.97.2.21 void ADC_SEQ_DMAConfig ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel )
```

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из ADC_SEQ_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 472

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

```
6.97.2.22 FlagStatus ADC_SEQ_DMAErrorStatus ( ADC_SEQ_Module_TypeDef
ADC_SEQ_Module )
```

Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущие состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 505

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
6.97.2.23 void ADC_SEQ_DMAErrorStatusClear ( ADC_SEQ_Module_TypeDef
ADC_SEQ_Module )
```

Сброс статуса ошибки DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 530

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.24 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 976

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.25 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1001

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.26 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 936

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.27 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 961

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.28 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 898

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.29 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение результата измерений из буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 880

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
6.97.2.30 uint32_t ADC_SEQ_GetFIFOLoad ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)
```

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 916

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.31 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

ITCount	
---------	--

См. определение в файле niietcm4_adc.c строка 750

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.32 void ADC_SEQ_Init (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 418

Перекрестные ссылки ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent, ADC_SEQ_Init_TypeDef::ADC_SEQ_SWReqEn, IS_ADC_CHANNEL, IS_ADC_DC, IS_ADC_SEQ_CONVERSION_COUNT, IS_ADC_SEQ_CONVERSION_DELAY, IS_ADC_SEQ_MODULE, IS_ADC_SEQ_START_EVENT и IS_FUNCTIONAL_STATE.

6.97.2.33 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
FunctionalState State)

Включение прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 697

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

```
6.97.2.34 void ADC_SEQ_ITConfig ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst )
```

Настройка вызова прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ ITRate	Значение количества перезапусков модулей АЦП секвенсором после которого генерируется прерывание. Параметр принимает любое значение из диапазона 1 - 256.
ADC_SEQ_↔ ITCountSEQRst	Разрешение сброса счетчика прерываний по запуску секвенсора. Если запретить, то счетчик можно будет сбрасывать только программно через ADC_SEQ_IT↔ CountRst . Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 725

Перекрестные ссылки IS_ADC_SEQ_IT_RATE, IS_ADC_SEQ_MODULE и IS_FUNCTIONAL↔
L_STATE.

```
6.97.2.35 void ADC_SEQ_ITCountRst ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module )
```

Сброс счетчика прерываний секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 768

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
6.97.2.36 FlagStatus ADC_SEQ_ITMaskedStatus ( ADC_SEQ_Module_TypeDef
ADC_SEQ_Module )
```

Проверка флагов маскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 807

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.37 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 782

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.38 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 832

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.97.2.39 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которую необходимо проинициализировать.
-------------------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 453

Перекрестные ссылки ADC_Channel_None, ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_D↔C_None, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::AD↔C_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef↔::ADC_SEQ_StartEvent, ADC_SEQ_StartEvent_SWReq и ADC_SEQ_Init_TypeDef::ADC_SE↔Q_SWReqEn.

6.97.2.40 void ADC_SEQ_SWReq ()

Программный запуск измерений всех разрешенных секвенсоров.

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 868

6.97.2.41 void ADC_StructInit (ADC_Init_TypeDef * ADC_InitStruct)

Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

Аргументы

ADC_Init↔Struct	Указатель на структуру типа ADC_Init_TypeDef , которую необходимо проинициализировать.
-----------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 283

Перекрестные ссылки ADC_Init_TypeDef::ADC_Average, ADC_Average_Disable, ADC_Init_↔TypeDef::ADC_Measure_A, ADC_Init_TypeDef::ADC_Measure_B, ADC_Measure_Single, ADC↔_Init_TypeDef::ADC_Mode, ADC_Mode_Powerdown, ADC_Init_TypeDef::ADC_Phase, ADC_↔Init_TypeDef::ADC_Resolution и ADC_Resolution_12bit.

6.98 BOOTFLASH

Драйвер для загрузочной флеш-памяти.

Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.98.1 Подробное описание

Драйвер для загрузочной флеш-памяти.

Внимание

Для контроллера K1921BK01T необходимо вызывать функции записи и стирания только из другой функции, расположенной в ОЗУ.

Общий вид процесса инициализации:

- обязательно инициализируем контроллер загрузочной памяти [BOOTFLASH_Init\(\)](#), передав в функцию значение текущей системной частоты;
- включаем прерывание если необходимо - [BOOTFLASH_ITCmd](#);
- контроллер загрузочной флеш-памяти готов к работе.

Более подробно инициализация и использование BOOTFLASH показаны в приложенных к драйверу примерах.

6.99 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.99.1 Подробное описание

6.100 Приватные константы

6.101 Приватные функции

Функции

- void [BOOTFLASH_Init](#) (uint32_t SysClkFreq)
Инициализирует тайминги доступа для контроллера загрузочной флеш.
- [BOOTFLASH_Status_TypeDef](#) [BOOTFLASH_OperationStatus](#) ()
Статус работы контроллера загрузочной флеш.
- void [BOOTFLASH_OperationStatusClear](#) ()
Очищает статус работы контроллера загрузочной флеш.
- void [BOOTFLASH_FullErase](#) ()
Полная очистка основной области загрузочной флеш.
- void [BOOTFLASH_Write](#) (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.
- void [BOOTFLASH_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы основной области загрузочной флеш.
- void [BOOTFLASH_Info_Write](#) (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.
- void [BOOTFLASH_Info_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы информационной области загрузочной флеш.
- void [BOOTFLASH_ITCmd](#) ([FunctionalState](#) State)
Включение прерывания по завершению чтения/записи/стирания.

6.101.1 Подробное описание

6.101.2 Функции

6.101.2.1 void [BOOTFLASH_FullErase](#) ()

Полная очистка основной области загрузочной флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле `niietcm4_bootflash.c` строка 112

Перекрестные ссылки [BOOTFLASH_MAGIC_KEY](#).

6.101.2.2 void [BOOTFLASH_Info_PageErase](#) (uint32_t PageNum)

Стирание указанной страницы информационной области загрузочной флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 179

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_↔ BOOTFLASH_INFO_PAGE_NUM.

6.101.2.3 void BOOTFLASH_Info_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 163

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.101.2.4 void BOOTFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера загрузочной флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
------------	---------------------------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 75

6.101.2.5 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.101.2.6 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из BOOTFLASH_H_Status_TypeDef .
--------	---

См. определение в файле niietcm4_bootflash.c строка 88

6.101.2.7 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 102

6.101.2.8 void BOOTFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области загрузочной флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 144

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_↵
BOOTFLASH_PAGE_NUM.

6.101.2.9 void BOOTFLASH_Write (uint32_t Address, uint32_t Data0, uint32_t Data1,
uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 128

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.102 CAP

Драйвер для блоков захвата

Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

6.102.1 Подробное описание

Драйвер для блоков захвата

Драйвер разбит на 3 субмодуля: [Конфигурация](#), [Режим ШИМ](#), [Режим захвата](#).

Вне зависимости от желаемого режима работы, сначала необходимо настроить блок в целом: [CAP_P_Init](#). Затем выполнить настройку на нужный режим: [CAP_PWM_Init](#) или [CAP_Capture_Init](#).

6.103 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.103.1 Подробное описание

6.104 Приватные константы

6.105 Приватные функции

Функции

- void [CAP_DeInit](#) (NT_CAP_TypeDef *CAPx)
Устанавливает все регистры блока захвата значениями по умолчанию.
- void [CAP_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_Init_TypeDef](#) *CAP_InitStruct)
Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.
- void [CAP_SyncCmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение синхронизации.
- void [CAP_StructInit](#) ([CAP_Init_TypeDef](#) *CAP_InitStruct)
Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.
- void [CAP_TimerCmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение работы таймера, выбранного блока захвата.
- void [CAP_SetTimer](#) (NT_CAP_TypeDef *CAPx, uint32_t TimerVal)
Установка текущего значения счетчика напрямую.
- void [CAP_SetShadowTimer](#) (NT_CAP_TypeDef *CAPx, uint32_t TimerVal)
Установка теневого значения таймера для отложенной записи.
- uint32_t [CAP_GetTimer](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения таймера.
- uint32_t [CAP_GetShadowTimer](#) (NT_CAP_TypeDef *CAPx)
Получение отложенного значения таймера.
- void [CAP_SwSync](#) (NT_CAP_TypeDef *CAPx)
Проведение программной синхронизации.
- void [CAP_PWM_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_PWM_Init_TypeDef](#) *CAP_PWM_↵
M_InitStruct)
Инициализирует режим ШИМ блока CAPx согласно параметрам структуры CAP_PWM_Init↵
Struct.
- void [CAP_PWM_StructInit](#) ([CAP_PWM_Init_TypeDef](#) *CAP_PWM_InitStruct)
Заполнение каждого члена структуры CAP_PWM_InitStruct значениями по умолчанию.
- void [CAP_PWM_SetPeriod](#) (NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)
Установка значения периода ШИМ.
- void [CAP_PWM_SetCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)
Установка значения сравнения ШИМ.
- void [CAP_PWM_SetShadowPeriod](#) (NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)
Установка значения периода ШИМ для отложенной записи.
- void [CAP_PWM_SetShadowCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)
Установка значения сравнения ШИМ для отложенной записи.
- uint32_t [CAP_PWM_GetPeriod](#) (NT_CAP_TypeDef *CAPx)
Получение текущего периода ШИМ.
- uint32_t [CAP_PWM_GetCompare](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения сравнения ШИМ.
- uint32_t [CAP_PWM_GetShadowPeriod](#) (NT_CAP_TypeDef *CAPx)
Получение отложенного значения периода ШИМ.
- uint32_t [CAP_PWM_GetShadowCompare](#) (NT_CAP_TypeDef *CAPx)
Получение отложенного значения сравнения ШИМ.
- void [CAP_Capture_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_Capture_Init_TypeDef](#) *CAP_↵
Capture_InitStruct)
Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_↵
InitStruct.
- void [CAP_Capture_StructInit](#) ([CAP_Capture_Init_TypeDef](#) *CAP_Capture_InitStruct)

- Заполнение каждого члена структуры CAP_Capture_InitStruct значениями по умолчанию.
- void CAP_Capture_Cmd (NT_CAP_TypeDef *CAPx, FunctionalState State)
Разрешение захвата для выбранного блока захвата.
 - void CAP_Capture_SetCap0 (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 0.
 - void CAP_Capture_SetCap1 (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 1.
 - void CAP_Capture_SetCap2 (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 2.
 - void CAP_Capture_SetCap3 (NT_CAP_TypeDef *CAPx, uint32_t Value)
Установка значения регистра захвата 3.
 - uint32_t CAP_Capture_GetCap0 (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 0.
 - uint32_t CAP_Capture_GetCap1 (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 1.
 - uint32_t CAP_Capture_GetCap2 (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 2.
 - uint32_t CAP_Capture_GetCap3 (NT_CAP_TypeDef *CAPx)
Получение текущего значения из регистра захвата 3.
 - void CAP_ITCmd (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)
Разрешение работы прерывания выбранного блока захвата.
 - void CAP_ITForceCmd (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)
Принудительный вызов прерывания выбранного блока захвата.
 - FlagStatus CAP_ITStatus (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)
Чтение статуса флага источника прерывания выбранного блока захвата.
 - void CAP_ITStatusClear (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)
Сброс флагов источников прерываний выбранного блока захвата.
 - FlagStatus CAP_ITPendStatus (NT_CAP_TypeDef *CAPx)
Чтение статуса прерывания выбранного блока захвата.
 - void CAP_ITPendClear (NT_CAP_TypeDef *CAPx)
Сброс флага прерывания выбранного блока захвата.

6.105.1 Подробное описание

6.105.2 Функции

6.105.2.1 void CAP_Capture_Cmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение захвата для выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 444

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

6.105.2.2 uint32_t CAP_Capture_GetCap0 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 0.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 519

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.3 uint32_t CAP_Capture_GetCap1 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 1.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 532

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.4 uint32_t CAP_Capture_GetCap2 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 2.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 545

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.5 uint32_t CAP_Capture_GetCap3 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 3.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 558

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.6 void CAP_Capture_Init (NT_CAP_TypeDef * CAPx, CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)

Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_↔ Capture_Init↔ Struct	Указатель на структуру типа CAP_Capture_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 386

Перекрестные ссылки `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`, `CAP_↔
Capture_Init_TypeDef::CAP_Capture_PolarityEvent1`, `CAP_Capture_Init_TypeDef::CAP_↔
Capture_PolarityEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`, `CAP_↔
Capture_Init_TypeDef::CAP_Capture_Prescale`, `CAP_Capture_Init_TypeDef::CAP_Capture_↔
RstEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1`, `CAP_Capture_Init_Type↔
Def::CAP_Capture_RstEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3`, `CAP_↔
Capture_Init_TypeDef::CAP_Capture_StopVal`, `CAP_Capture_Init_TypeDef::CAP_CaptureMode`, `IS_CAP_ALL_PERIPH`, `IS_CAP_CAPTURE_MODE`, `IS_CAP_CAPTURE_POLARITY`, `IS_↔
CAP_CAPTURE_PRESCALE`, `IS_CAP_CAPTURE_STOP_VAL` и `IS_FUNCTIONAL_STATE`.

6.105.2.7 `void CAP_Capture_SetCap0 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 0.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 464

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.8 `void CAP_Capture_SetCap1 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 1.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 478

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.9 `void CAP_Capture_SetCap2 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 2.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 492

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.10 `void CAP_Capture_SetCap3 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 3.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 506

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.11 `void CAP_Capture_StructInit (CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)`

Заполнение каждого члена структуры `CAP_Capture_InitStruct` значениями по умолчанию.

Аргументы

CAP_Capture_InitStruct	Указатель на структуру типа <code>CAP_Capture_Init_TypeDef</code> , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 421

Перекрестные ссылки `CAP_Capture_Mode_Single`, `CAP_Capture_Polarity_PosEdge`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`, `CAP_Capture_Init_TypeDef::CAP_Capture_Prescale`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3`, `CAP_Capture_Init_TypeDef::CAP_Capture_StopVal` и `CAP_Capture_Init_TypeDef::CAP_CaptureMode`.

6.105.2.12 `void CAP_DeInit (NT_CAP_TypeDef * CAPx)`

Устанавливает все регистры блока захвата значениями по умолчанию.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 68

Перекрестные ссылки IS_CAP_ALL_PERIPH, RCC_PeriphRst_CAP0, RCC_PeriphRst_CAP1, RCC_PeriphRst_CAP2, RCC_PeriphRst_CAP3, RCC_PeriphRst_CAP4, RCC_PeriphRst_CAP5 и RCC_PeriphRstCmd().

6.105.2.13 uint32_t CAP_GetShadowTimer (NT_CAP_TypeDef * CAPx)

Получение отложенного значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 218

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.14 uint32_t CAP_GetTimer (NT_CAP_TypeDef * CAPx)

Получение текущего значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 205

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.15 void CAP_Init (NT_CAP_TypeDef * CAPx, CAP_Init_TypeDef * CAP_InitStruct)

Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef, которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 111

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Init_TypeDef::CAP_Mode, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut, IS_CAP_ALL_PERIPH, IS_CAP_HALT, IS_CAP_MODE и IS_CAP_SYNC_OUT.

6.105.2.16 void CAP_ITCmd (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource, FunctionalState State)

Разрешение работы прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT← Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 575

Перекрестные ссылки IS_CAP_ALL_PERIPH, IS_CAP_IT_SOURCE и IS_FUNCTIONAL_STATE.

6.105.2.17 void CAP_ITForceCmd (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Принудительный вызов прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT← Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 599

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

6.105.2.18 void CAP_ITPendClear (NT_CAP_TypeDef * CAPx)

Сброс флага прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 680

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.19 FlagStatus CAP_ITPendStatus (NT_CAP_TypeDef * CAPx)

Чтение статуса прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 656

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.20 FlagStatus CAP_ITStatus (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Чтение статуса флага источника прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 615

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE_SINGLE.

6.105.2.21 void CAP_ITStatusClear (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Сброс флагов источников прерываний выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 642

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

6.105.2.22 uint32_t CAP_PWM_GetCompare (NT_CAP_TypeDef * CAPx)

Получение текущего значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле niietcm4_cap.c строка 345

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.23 `uint32_t CAP_PWM_GetPeriod (NT_CAP_TypeDef * CAPx)`

Получение текущего периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле `niietcm4_cap.c` строка 332

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.24 `uint32_t CAP_PWM_GetShadowCompare (NT_CAP_TypeDef * CAPx)`

Получение отложенного значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле `niietcm4_cap.c` строка 371

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.25 `uint32_t CAP_PWM_GetShadowPeriod (NT_CAP_TypeDef * CAPx)`

Получение отложенного значения периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле `niietcm4_cap.c` строка 358

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.26 `void CAP_PWM_Init (NT_CAP_TypeDef * CAPx, CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)`

Инициализирует режим ШИМ блока CAPx согласно параметрам структуры CAP_PWM_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_PWM_InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_cap.c` строка 246

Перекрестные ссылки `CAP_PWM_Init_TypeDef::CAP_PWM_Compare`, `CAP_PWM_Init_TypeDef::CAP_PWM_Period`, `CAP_PWM_Init_TypeDef::CAP_PWM_Polarity`, `CAP_PWM_SetCompare()`, `CAP_PWM_SetPeriod()`, `IS_CAP_ALL_PERIPH` и `IS_CAP_PWM_POLARITY`.

6.105.2.27 void CAP_PWM_SetCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 291

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

6.105.2.28 void CAP_PWM_SetPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 277

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

6.105.2.29 void CAP_PWM_SetShadowCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 319

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.30 void CAP_PWM_SetShadowPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 305

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.31 `void CAP_PWM_StructInit (CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)`

Заполнение каждого члена структуры `CAP_PWM_InitStruct` значениями по умолчанию.

Аргументы

CAP_PWM_↔ _InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef , которую необходимо проинициализировать.
--------------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 263

Перекрестные ссылки `CAP_PWM_Init_TypeDef::CAP_PWM_Compare`, `CAP_PWM_Init_↔
TypeDef::CAP_PWM_Period`, `CAP_PWM_Init_TypeDef::CAP_PWM_Polarity` и `CAP_PWM_↔
Polarity_Pos`.

6.105.2.32 `void CAP_SetShadowTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)`

Установка теневого значения таймера для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 192

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

6.105.2.33 `void CAP_SetTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)`

Установка текущего значения счетчика напрямую.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

--

Нет

См. определение в файле niietcm4_cap.c строка 178

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.34 void CAP_StructInit (CAP_Init_TypeDef * CAP_InitStruct)

Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.

Аргументы

CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef, которую необходимо проинициализировать.
----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 147

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Halt_Stop, CAP_Init_TypeDef::CAP_Mode, CAP_Mode_Capture, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut и CAP_SyncOut_Bypass.

6.105.2.35 void CAP_SwSync (NT_CAP_TypeDef * CAPx)

Проведение программной синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 231

Перекрестные ссылки IS_CAP_ALL_PERIPH.

6.105.2.36 void CAP_SyncCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
State	Выбор состояния. Параметр принимает любое значение из FunctionalState.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 132

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

6.105.2.37 void CAP_TimerCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение работы таймера, выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 163

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

6.106 DMA

Драйвер для работы с контроллером прямого доступа памяти.

Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.106.1 Подробное описание

Драйвер для работы с контроллером прямого доступа памяти.

Внимание

Для работы драйвер требует наличия размещенной во внутреннем ОЗУ структуры типа [DMA_ConfigData_TypeDef](#) - структуры первичных и альтернативных управляющих данных каналов. Размер этой структуры составляет 1кБ и она должна быть обязательно выравнена по 1024 байтам в адресном пространстве.

Общий вид процесса инициализации:

- создаем структуру типа [DMA_ConfigData_TypeDef](#);
- передаем адрес этой структуры контроллеру DMA - [DMA_BasePtrConfig](#);
- инициализируем необходимые каналы ([Инициализация каналов DMA](#));
- инициализируем контроллер DMA ([Инициализация контроллера DMA](#) или через отдельные функции [Конфигурация контроллера DMA](#));
- DMA готов к работе.

Более подробно инициализация и использование DMA показаны в приложенных к драйверу примерах.

6.107 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.107.1 Подробное описание

6.108 Приватные константы

Группы

- [Начальные значения регистров](#)

6.108.1 Подробное описание

6.109 Начальные значения регистров

6.110 Приватные функции

Функции

- void [DMA_ChannelDeInit](#) ([DMA_Channel_TypeDef](#) *DMA_Channel)
Деинициализация канала DMA.
- void [DMA_ChannelInit](#) ([DMA_Channel_TypeDef](#) *DMA_Channel, [DMA_ChannelInit_TypeDef](#) *DMA_ChannelInitStruct)
Инициализация канала DMA.
- void [DMA_ChannelStructInit](#) ([DMA_ChannelInit_TypeDef](#) *DMA_ChannelInitStruct)
Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.
- void [DMA_DeInit](#) ()
Деинициализация контроллера DMA.
- void [DMA_Init](#) ([DMA_Init_TypeDef](#) *DMA_InitStruct)
Инициализация контроллера DMA.
- void [DMA_StructInit](#) ([DMA_Init_TypeDef](#) *DMA_InitStruct)
Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.
- void [DMA_BasePtrConfig](#) (uint32_t BasePtr)
Установка базового адреса управляющих каналов.
- void [DMA_ProtectionConfig](#) ([DMA_Protect_TypeDef](#) *DMA_Protection)
Управление защитой шины при обращении DMA к управляющим данным.
- void [DMA_MasterEnableCmd](#) ([FunctionalState](#) State)
Разрешения работы контроллера DMA.
- void [DMA_SWRequestCmd](#) (uint32_t DMA_Channel)
Программный запрос на осуществление передач DMA по выбранным каналам.
- void [DMA_UseBurstCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Установка пакетного обмена каналов DMA.
- void [DMA_ReqMaskCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Маскирование каналов DMA.
- void [DMA_ChannelEnableCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Активация каналов DMA.
- void [DMA_PrmAltCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Установка первичной/альтернативной управляющей структуры каналов DMA.
- void [DMA_HighPriorityCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Установка высокого приоритета каналов DMA.
- [DMA_State_TypeDef](#) [DMA_StateStatus](#) ()
Доступ к текущему конечного автомата контроллера DMA.
- [FunctionalState](#) [DMA_MasterEnableStatus](#) ()
Состояние контроллера DMA.
- [FunctionalState](#) [DMA_WaitOnReqStatus](#) (uint32_t DMA_Channel)
Показывает поддерживает ли канал одиночные SREQ запросы.
- [OperationStatus](#) [DMA_ErrorStatus](#) ()
Показывает наличие ошибки на шине.
- void [DMA_ClearErrorStatus](#) ()
Сброс флага ошибки на шине.

6.110.1 Подробное описание

6.110.2 Функции

6.110.2.1 void DMA_BasePtrConfig (uint32_t BasePtr)

Установка базового адреса управляющих каналов.

Аргументы

BasePtr	Значение базового адреса.
---------	---------------------------

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 231

6.110.2.2 `void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)`

Деинициализация канала DMA.

Аргументы

DMA_Channel	Указатель на структуру типа DMA_Channel_TypeDef , которая содержит конфигурационную информацию канала.
-------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 87

Перекрестные ссылки `DMA_Channel_TypeDef::CHANNEL_CFG`, `DMA_Channel_TypeDef::DST_DATA_END` и `DMA_Channel_TypeDef::SRC_DATA_END`.

6.110.2.3 `void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)`

Активация каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 373

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.110.2.4 `void DMA_ChannelInit (DMA_Channel_TypeDef * DMA_Channel, DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)`

Инициализация канала DMA.

Аргументы

DMA_Channel	Непосредственно сама структура канала.
DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInitStruct_TypeDef , которая содержит конфигурационную информацию канала.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 102

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Channel_TypeDef::CHANNEL_CFG_bit, CHANNEL_CFG_bits::CYCLE_CTRL, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal, DMA_Channel_TypeDef::DST_DATA_END, CHANNEL_CFG_bits::DST_INC, CHANNEL_CFG_bits::DST_PROT_BUFFERABLE, CHANNEL_CFG_bits::DST_PROT_CACHEABLE, CHANNEL_CFG_bits::DST_PROT_PRIVILEGED, CHANNEL_CFG_bits::DST_SIZE, IS_DMA_ARBITRATION_RATE, IS_DMA_DATA_INC, IS_DMA_DATA_SIZE, IS_DMA_MODE, IS_DMA_TRANSFERS_TOTAL, IS_FUNCTIONAL_STATE, CHANNEL_CFG_bits::N_MINUS_1, CHANNEL_CFG_bits::NEXT_USEBURST, DMA_Protect_TypeDef::PRIVELGED, CHANNEL_CFG_bits::R_POWER, DMA_Channel_TypeDef::SRC_DATA_END, CHANNEL_CFG_bits::SRC_INC, CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE, CHANNEL_CFG_bits::SRC_PROT_CACHEABLE, CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED и CHANNEL_CFG_bits::SRC_SIZE.

6.110.2.5 void DMA_ChannelStructInit (DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

Аргументы

DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которую необходимо проинициализировать.
-----------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 146

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ArbitrationRate_1, DMA_DataInc_Disable, DMA_DataSize_8, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_Mode_Disable, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal и DMA_Protect_TypeDef::PRIVELGED.

6.110.2.6 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 524

6.110.2.7 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 174

6.110.2.8 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

Status	Одно из значений OperationStatus: <ul style="list-style-type: none"> • OK - ошибок не было; • ERROR - произошла ошибка.
--------	---

См. определение в файле niietcm4_dma.c строка 503

6.110.2.9 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

6.110.2.10 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

Прежде чем инициализировать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать базовый адрес управляющей структуры с помощью [DMA_BasePtrConfig](#).

Аргументы

DMA_Init↔ Struct	Указатель на структуру типа DMA_Init_TypeDef , которая содержит конфигурационную информацию.
---------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 195

Перекрестные ссылки [DMA_Init_TypeDef::DMA_Channel](#), [DMA_Init_TypeDef::DMA_Channel↔Enable](#), [DMA_ChannelEnableCmd\(\)](#), [DMA_Init_TypeDef::DMA_HighPriority](#), [DMA_HighPriority↔Cmd\(\)](#), [DMA_Init_TypeDef::DMA_PrmAlt](#), [DMA_PrmAltCmd\(\)](#), [DMA_Init_TypeDef::DMA_↔Protection](#), [DMA_ProtectionConfig\(\)](#), [DMA_Init_TypeDef::DMA_ReqMask](#), [DMA_ReqMaskCmd\(\)](#), [DMA_Init_TypeDef::DMA_UseBurst](#) и [DMA_UseBurstCmd\(\)](#).

6.110.2.11 `void DMA_MasterEnableCmd (FunctionalState State)`

Разрешения работы контроллера DMA.

Внимание

Прежде чем включать DMA, необходимо проинициализировать каналы с помощью [DMA_↔ChannelInit](#) и сконфигурировать контроллер DMA через функцию инициализации [DMA_Init](#) или вручную - [Конфигурация контроллера DMA](#).

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 287

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.110.2.12 `FunctionalState DMA_MasterEnableStatus ()`

Состояние контроллера DMA.

Возвращаемые значения

Status	Текущее состояние контроллера DMA.
--------	------------------------------------

См. определение в файле `niietcm4_dma.c` строка 455

6.110.2.13 `void DMA_PrmAltCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 397

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.110.2.14 `void DMA_ProtectionConfig (DMA_Protect_TypeDef * DMA_Protection)`

Управление защитой шины при обращении DMA к управляющим данным.

Аргументы

<code>DMA_↔ Protection</code>	Структура, содержащая конфигурацию защиты. Параметр принимает структуру типа DMA_Protect_TypeDef .
-----------------------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 243

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHE↔
ABLE`, `IS_FUNCTIONAL_STATE` и `DMA_Protect_TypeDef::PRIVELGED`.

Используется в `DMA_Init()`.

6.110.2.15 `void DMA_ReqMaskCmd (uint32_t DMA_Channel, FunctionalState State)`

Маскирование каналов DMA.

Внимание

По маскированным каналам игнорируются запросы на передачи.

Аргументы

<code>DMA_Channel</code>	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 349

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.110.2.16 `DMA_State_TypeDef DMA_StateStatus ()`

Доступ к текущему конечного автомата контроллера DMA.

Возвращаемые значения

<code>State</code>	Текущее состояние конечного автомата.
--------------------	---------------------------------------

См. определение в файле `niietcm4_dma.c` строка 441

6.110.2.17 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

Аргументы

DMA_Init_↵ Struct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
----------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 212

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHE↵ABLE`, `DMA_Init_TypeDef::DMA_Channel`, `DMA_Init_TypeDef::DMA_ChannelEnable`, `DMA_↵Init_TypeDef::DMA_HighPriority`, `DMA_Init_TypeDef::DMA_PrmAlt`, `DMA_Init_TypeDef::DM↵A_Protection`, `DMA_Init_TypeDef::DMA_ReqMask`, `DMA_Init_TypeDef::DMA_UseBurst` и `DM↵A_Protect_TypeDef::PRIVELGED`.

6.110.2.18 `void DMA_SWRequestCmd (uint32_t DMA_Channel)`

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
-------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 308

Перекрестные ссылки `IS_DMA_CHANNEL`.

6.110.2.19 `void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка пакетного обмена каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 324

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.110.2.20 `FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)`

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

Status	Одно из значений FunctionalState: <ul style="list-style-type: none">• ENABLE - поддерживаются SREQ (как и блочные BREQ);• DISABLE - поддерживаются только блочные запросы BREQ.
--------	--

См. определение в файле `niietcm4_dma.c` строка 478

Перекрестные ссылки `IS_GET_DMA_CHANNEL`.

6.111 EXTMEM

Драйвер для интерфейса внешней памяти.

Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.111.1 Подробное описание

Драйвер для интерфейса внешней памяти.

Внимание

Драйвер позволяет управлять только внутренними настройками интерфейса внешней памяти. Порты ввода-вывода настраиваются отдельно через [GPIO](#) :
- [Инициализация и деинициализация](#).

Общий вид процесса инициализации:

- передаем параметры через структуру типа [EXTMEM_Init_TypeDef](#) в функцию [EXTMEM↵_Init](#);
- интерфейс внешней памяти готов к работе.

Более подробно инициализация и использование внешней памяти показаны в приложенных к драйверу примерах.

6.112 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.112.1 Подробное описание

6.113 Приватные константы

Группы

- [Начальные значения регистров](#)

6.113.1 Подробное описание

6.114 Начальные значения регистров

Макросы

- `#define EXT_MEM_CFG_Reset_Value ((uint32_t)0x80000007)`

6.114.1 Подробное описание

6.114.2 Макросы

6.114.2.1 `#define EXT_MEM_CFG_Reset_Value ((uint32_t)0x80000007)`

Значение по сбросу регистра EXT_MEM_CFG

См. определение в файле `niietcm4_extmem.c` строка 64

Используется в `EXTMEM_DeInit()`.

6.115 Приватные функции

Функции

- void [EXTMEM_Init](#) ([EXTMEM_Init_TypeDef](#) *[EXTMEM_InitStruct](#))
Инициализирует внешнюю память согласно параметрам структуры [EXTMEM_InitStruct](#).
- void [EXTMEM_StructInit](#) ([EXTMEM_Init_TypeDef](#) *[EXTMEM_InitStruct](#))
Заполнение каждого члена структуры [EXTMEM_InitStruct](#) значениями по умолчанию.
- void [EXTMEM_DeInit](#) ()
Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.

6.115.1 Подробное описание

6.115.2 Функции

6.115.2.1 void [EXTMEM_DeInit](#) ()

Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле [nietcm4_extmem.c](#) строка 121

Перекрестные ссылки [EXT_MEM_CFG_Reset_Value](#).

6.115.2.2 void [EXTMEM_Init](#) ([EXTMEM_Init_TypeDef](#) * [EXTMEM_InitStruct](#))

Инициализирует внешнюю память согласно параметрам структуры [EXTMEM_InitStruct](#).

Аргументы

EXTMEM_InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
-----------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле [nietcm4_extmem.c](#) строка 85

Перекрестные ссылки [IS_EXTMEM_CE_MASK](#), [IS_EXTMEM_READ_WAITSTATE](#), [IS_EXTMEM_RW_WAITSTATE](#), [IS_EXTMEM_WIDTH](#) и [IS_EXTMEM_WRITE_WAITSTATE](#).

6.115.2.3 void [EXTMEM_StructInit](#) ([EXTMEM_Init_TypeDef](#) * [EXTMEM_InitStruct](#))

Заполнение каждого члена структуры [EXTMEM_InitStruct](#) значениями по умолчанию.

Аргументы

EXTMEM_InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
-----------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_extmem.c` строка 107

Перекрестные ссылки `EXTMEM_Init_TypeDef::CEMask`, `EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState`, `EXTMEM_ReadWaitState_8`, `EXTMEM_Init_TypeDef::EXTMEM_RWWaitState`, `EXTMEM_RWWaitState_1`, `EXTMEM_Init_TypeDef::EXTMEM_Width`, `EXTMEM_Width_16bit`, `EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState` и `EXTMEM_WriteWaitState_1`.

6.116 GPIO

Драйвер для управления портами ввода-вывода.

Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

6.116.1 Подробное описание

Драйвер для управления портами ввода-вывода.

Внимание

Необходимо учитывать алгоритм выбора альтернативной функции для входа какой-либо периферии. Если одна и та же функция периферии расположена на разных портах под разными альтернативными функциями (например RX у UART), то периферия будет выбирать функцию с наименьшим номером. Поэтому чтобы выбирать функции отличные от первой, необходимо пины с альтернативными функциями с меньшими номерами сконфигурировать на альтернативную функцию с номером большим либо равным желаемой. Подробнее этот вопрос раскрыт в ТО, а также показан в примерах работы того же UART.

Общий вид процесса инициализации:

- инициализируем необходимые модули порты и их пины ([Инициализация и деинициализация](#));
- (опционально) настраиваем фильтрацию входного сигнала ([Фильтрация](#));
- (опционально) настраиваем и включаем прерывания ([Прерывания](#));
- порты ввода-вывода готовы к работе.

Более подробно инициализация и использование портов показаны в приложенных к драйверу примерах. Примеры использования альтернативных функций можно найти в примерах работы с другой периферией, которая использует порты для функционирования.

6.117 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.117.1 Подробное описание

6.118 Приватные константы

Группы

- [Начальные значения регистров](#)
- [Маски портов](#)

6.118.1 Подробное описание

6.119 Начальные значения регистров

Макросы

- `#define GPIO_DATAOUT_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIODEN0_Reset_Value ((uint32_t)0x00020062)`
- `#define GPIO_GPIODEN1_Reset_Value ((uint32_t)0x08000000)`
- `#define GPIO_GPIODEN2_Reset_Value ((uint32_t)0x00000400)`
- `#define GPIO_GPIODEN3_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOODCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOODSCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOPUCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOSEx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQEx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQMx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOPCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQPx_Reset_Value ((uint32_t)0x00000000)`

6.119.1 Подробное описание

6.119.2 Макросы

6.119.2.1 `#define GPIO_DATAOUT_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра DATAOUT

См. определение в файле `niietcm4_gpio.c` строка 72

Используется в `GPIO_DeInit()`.

6.119.2.2 `#define GPIO_GPIODEN0_Reset_Value ((uint32_t)0x00020062)`

Значение по сбросу регистра GPIODEN0

См. определение в файле `niietcm4_gpio.c` строка 73

Используется в `GPIO_DeInit()`.

6.119.2.3 `#define GPIO_GPIODEN1_Reset_Value ((uint32_t)0x08000000)`

Значение по сбросу регистра GPIODEN1

См. определение в файле `niietcm4_gpio.c` строка 74

Используется в `GPIO_DeInit()`.

6.119.2.4 `#define GPIO_GPIODEN2_Reset_Value ((uint32_t)0x00000400)`

Значение по сбросу регистра GPIODEN2

См. определение в файле `niietcm4_gpio.c` строка 75

Используется в `GPIO_DeInit()`.

6.119.2.5 `#define GPIO_GPIODEN3_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIODEN3

См. определение в файле `niietcm4_gpio.c` строка 76

Используется в `GPIO_DeInit()`.

6.119.2.6 `#define GPIO_GPIOODCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOODCTLx

См. определение в файле `niietcm4_gpio.c` строка 77

Используется в `GPIO_DeInit()`.

6.119.2.7 `#define GPIO_GPIOODSCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOODSCTLx

См. определение в файле `niietcm4_gpio.c` строка 78

Используется в `GPIO_DeInit()`.

6.119.2.8 `#define GPIO_GPIOPCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOPCTLx

См. определение в файле `niietcm4_gpio.c` строка 83

Используется в `GPIO_DeInit()`.

6.119.2.9 `#define GPIO_GPIOPUCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOPUCTLx

См. определение в файле `niietcm4_gpio.c` строка 79

Используется в `GPIO_DeInit()`.

6.119.2.10 `#define GPIO_GPIOQEx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOQEx

См. определение в файле `niietcm4_gpio.c` строка 81

Используется в `GPIO_DeInit()`.

6.119.2.11 `#define GPIO_GPIOQMx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOQMx

См. определение в файле `niietcm4_gpio.c` строка 82

Используется в `GPIO_DeInit()`.

6.119.2.12 `#define GPIO_GPIOQPx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOQPx

См. определение в файле `niietcm4_gpio.c` строка 84

Используется в `GPIO_DeInit()`.

6.119.2.13 `#define GPIO_GPIOSEx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOSEx

См. определение в файле `niietcm4_gpio.c` строка 80

Используется в `GPIO_DeInit()`.

6.120 Маски портов

Макросы

- `#define GPIO_Regs_A_C_E_G_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_B_D_F_H_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOA_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOB_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOC_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOD_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOE_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOF_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOG_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

6.120.1 Подробное описание

6.120.2 Макросы

6.120.2.1 `#define GPIO_Regs_A_C_E_G_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для нечетных портов

См. определение в файле `niietcm4_gpio.c` строка 94

Используется в `GPIO_DeInit()`.

6.120.2.2 `#define GPIO_Regs_B_D_F_H_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для четных портов

См. определение в файле `niietcm4_gpio.c` строка 95

Используется в `GPIO_DeInit()`.

6.120.2.3 `#define GPIO_Regs_GPIOA_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOA

См. определение в файле `niietcm4_gpio.c` строка 96

6.120.2.4 `#define GPIO_Regs_GPIOB_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOB

См. определение в файле `niietcm4_gpio.c` строка 97

6.120.2.5 `#define GPIO_Regs_GPIOC_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOC

См. определение в файле `niietcm4_gpio.c` строка 98

6.120.2.6 `#define GPIO_Regs_GPIOD_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOD

См. определение в файле `niietcm4_gpio.c` строка 99

6.120.2.7 `#define GPIO_Regs_GPIOE_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOE

См. определение в файле `niietcm4_gpio.c` строка 100

6.120.2.8 `#define GPIO_Regs_GPIOF_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOF

См. определение в файле `niietcm4_gpio.c` строка 101

6.120.2.9 `#define GPIO_Regs_GPIOG_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOG

См. определение в файле `niietcm4_gpio.c` строка 102

6.120.2.10 `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOH

См. определение в файле `niietcm4_gpio.c` строка 103

6.121 Приватные функции

Функции

- void **GPIO_DeInit** (NT_GPIO_TypeDef *GPIOx)
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- void **GPIO_AltFuncConfig** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **GPIO_AltFunc_TypeDef** GPIO_AltFunc)
- void **GPIO_Init** (NT_GPIO_TypeDef *GPIOx, **GPIO_Init_TypeDef** *GPIO_InitStruct)
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- void **GPIO_StructInit** (**GPIO_Init_TypeDef** *GPIO_InitStruct)
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.
- uint32_t **GPIO_ReadBit** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Чтение состояния выбранного пина.
- uint32_t **GPIO_Read** (NT_GPIO_TypeDef *GPIOx)
Чтение состояния выбранного порта GPIOx.
- uint32_t **GPIO_ReadMask** (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)
Чтение состояния выбранного порта GPIOx с использованием маски.
- void **GPIO_WriteBit** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **BitAction** BitVal)
Изменение состояния выбранного пина.
- void **GPIO_Write** (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx.
- void **GPIO_WriteMask** (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx с использованием маски.
- void **GPIO_SetBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Установка выбранных пинов.
- void **GPIO_ClearBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Сброс выбранных пинов.
- void **GPIO_ToggleBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Переключение выбранных пинов в противоположное состояние.
- void **GPIO_QualConfig** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **GPIO_QualMode_TypeDef** Mode, uint32_t SamplePeriod)
Настройка фильтра выбранных пинов.
- void **GPIO_QualCmd** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **FunctionalState** State)
Включение входных фильтров.
- void **GPIO_SyncCmd** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **FunctionalState** State)
Включение пересинхронизации входов через 2 триггера-защелки.
- void **GPIO_ITConfig** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **GPIO_IntType_TypeDef** IntType, **GPIO_IntPol_TypeDef** IntPol)
Настройка прерываний пинов.
- void **GPIO_ITCmd** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **FunctionalState** State)
Включение прерываний выбранных пинов.
- void **GPIO_ITStatusClear** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Очистка флагов прерываний выбранных пинов.

6.121.1 Подробное описание

6.121.2 Функции

6.121.2.1 void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niitcm4_gpio.c строка 626

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.121.2.2 void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)

Устанавливает все регистры выбранного GPIOx значениями по умолчанию.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Нет

См. определение в файле niitcm4_gpio.c строка 123

Перекрестные ссылки GPIO_DATAOUT_Reset_Value, GPIO_GPIODEN0_Reset_Value, GPIO_GPIODEN1_Reset_Value, GPIO_GPIODEN2_Reset_Value, GPIO_GPIODEN3_Reset_Value, GPIO_GPIOODCTLx_Reset_Value, GPIO_GPIOODSCTLx_Reset_Value, GPIO_GPIOPCTLx_Reset_Value, GPIO_GPIOPUCTLx_Reset_Value, GPIO_GPIOQEx_Reset_Value, GPIO_GPIOQMx_Reset_Value, GPIO_GPIOQPx_Reset_Value, GPIO_GPIOSEx_Reset_Value, GPIO_Regs_A_C_E_G_Mask, GPIO_Regs_B_D_F_H_Mask и IS_GPIO_ALL_PERIPH.

6.121.2.3 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niitcm4_gpio.c строка 331

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_In, GPIO_Dir_Out, GPIO_Init_TypeDef::GPIO_Load, GPIO_Load_16mA, GPIO_Load_8mA, GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Mode_IO, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_Dis, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_OutMode, GPIO_OutMode_OD, GPIO_OutMode_PP, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Init_TypeDef::GPIO_PullUp, GPIO_PullUp_Dis, GPIO_PullUp_En, IS_GPIO_ALL_PERIPH, IS_GPIO_ALT_FUNC, IS_GPIO_DIR, IS_GPIO_LOAD, IS_GPIO_MODE, IS_GPIO_OUT, IS_GPIO_OUT_MODE, IS_GPIO_PIN и IS_GPIO_PULLUP.

Используется в RCC_SysClkDiv2Out().

6.121.2.4 void GPIO_ITCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,
FunctionalState State)

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 913

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.121.2.5 `void GPIO_ITConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)`

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 876

Перекрестные ссылки `GPIO_IntPol_Neg`, `GPIO_IntPol_Pos`, `GPIO_IntType_Edge`, `GPIO_IntType_Level`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_INT_POL`, `IS_GPIO_INT_TYPE` и `IS_GPIO_PIN`.

6.121.2.6 `void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 938

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.121.2.7 `void GPIO_QualCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)`

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.121.2.8 void GPIO_QualConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePerod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_QualMode_TypeDef .
SamplePerod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 664

Перекрестные ссылки GPIO_QualMode_3sample, GPIO_QualMode_6sample, IS_GPIO_ALL_PERIPH, IS_GPIO_PIN, IS_GPIO_QUAL_MODE и IS_GPIO_QUAL_PERIOD.

6.121.2.9 uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx

См. определение в файле niietcm4_gpio.c строка 503

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.121.2.10 uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина	
---------------------------	--

См. определение в файле `niietcm4_gpio.c` строка 477

Перекрестные ссылки `Bit_CLEAR`, `Bit_SET`, `IS_GET_GPIO_PIN` и `IS_GPIO_ALL_PERIPH`.

6.121.2.11 `uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)`

Чтение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски чтения.

Возвращаемые значения

Состояние порта GPIOx с учетом маски	
--------------------------------------	--

См. определение в файле `niietcm4_gpio.c` строка 518

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

6.121.2.12 `void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 609

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.121.2.13 `void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)`

Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 456

Перекрестные ссылки `GPIO_InitTypeDef::GPIO_AltFunc`, `GPIO_AltFunc_1`, `GPIO_InitTypeDef::GPIO_Dir`, `GPIO_Dir_In`, `GPIO_InitTypeDef::GPIO_Load`, `GPIO_Load_8mA`, `GPIO_InitTypeDef::GPIO_Mode`, `GPIO_Mode_IO`, `GPIO_InitTypeDef::GPIO_Out`, `GPIO_Out_Dis`, `GPIO_InitTypeDef::GPIO_OutMode`, `GPIO_OutMode_PP`, `GPIO_InitTypeDef::GPIO_Pin`, `GPIO_Pin_All`, `GPIO_InitTypeDef::GPIO_PullUp` и `GPIO_PullUp_Dis`.

Используется в `RCC_SysClkDiv2Out()`.

6.121.2.14 `void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)`

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

<code>GPIOx</code>	выбор порта, где x лежит в диапазоне A..H.
<code>GPIO_Pin</code>	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 814

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.121.2.15 `void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Переключение выбранных пинов в противоположное состояние.

Аргументы

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне A..H.
<code>GPIO_Pin</code>	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 643

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.121.2.16 `void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)`

Изменение состояния выбранного порта `GPIOx`.

Аргументы

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне A..H.
--------------------	--

PortVal	Значение которое будет записано.
---------	----------------------------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.121.2.17 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 546

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN, IS_GPIO_ALL_PERIPH и IS_GPIO_BIT_ACTION.

6.121.2.18 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 586

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.122 RCC

Драйвер для работы с тактированием и сбросом периферийных блоков.

Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.122.1 Подробное описание

Драйвер для работы с тактированием и сбросом периферийных блоков.

6.123 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.123.1 Подробное описание

6.124 Приватные константы

Группы

- [Начальные значения регистров](#)

6.124.1 Подробное описание

6.125 Начальные значения регистров

Макросы

- `#define RCC_PLL_CTRL_Reset_Value ((uint32_t)0x00000000)`
- `#define RCC_PLL_OD_Reset_Value ((uint32_t)0x00000000)`
- `#define RCC_PLL_NR_Reset_Value ((uint32_t)0x00000000)`
- `#define RCC_PLL_NF_Reset_Value ((uint32_t)0x00000000)`

6.125.1 Подробное описание

6.125.2 Макросы

6.125.2.1 `#define RCC_PLL_CTRL_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_CTRL

См. определение в файле `niietcm4_rcc.c` строка 54

Используется в `RCC_PLLDeInit()`.

6.125.2.2 `#define RCC_PLL_NF_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_NF

См. определение в файле `niietcm4_rcc.c` строка 57

Используется в `RCC_PLLDeInit()`.

6.125.2.3 `#define RCC_PLL_NR_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_NR

См. определение в файле `niietcm4_rcc.c` строка 56

Используется в `RCC_PLLDeInit()`.

6.125.2.4 `#define RCC_PLL_OD_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_OD

См. определение в файле `niietcm4_rcc.c` строка 55

Используется в `RCC_PLLDeInit()`.

6.126 Приватные функции

Функции

- `uint32_t RCC_WaitClkChange (RCC_SysClk_TypeDef RCC_SysClk)`
Процедура ожидания смены источника тактового сигнала
- `void RCC_SysClkDiv2Out (FunctionalState State)`
Включение генерации тактового сигнала с частотой равной половине системной на выводе H[0].
Функция использует драйвер GPIO для настройки выхода.
- `OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)`
Автоматическая конфигурация PLL для получения желаемой системной частоты.
- `void RCC_PLLInit (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)`
Инициализирует PLL согласно параметрам структуры `RCC_PLLInit_Struct`.
- `void RCC_PLLStructInit (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)`
Заполнение каждого члена структуры `RCC_PLLInit_Struct` значениями по умолчанию.
- `void RCC_PLLDeInit ()`
Устанавливает все регистры PLL значениями по умолчанию.
- `void RCC_PLLPowerDownCmd (FunctionalState State)`
Управление режимом PowerDown PLL.
- `void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)`
Включение тактирования выбранного блока периферии.
- `OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)`
Выбор источника для системного тактового сигнала.
- `RCC_SysClk_TypeDef RCC_SysClkStatus ()`
Текущий источник системного тактового сигнала.
- `void RCC_USBClkConfig (RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)`
Настройка источника тактового сигнала для USB.
- `void RCC_USBClkCmd (FunctionalState State)`
Включение тактирования USB.
- `void RCC_UARTClkSel (NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)`
Настройка источника тактового сигнала для выбранного UART.
- `void RCC_UARTClkDivConfig (NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)`
Настройка делителя тактового сигнала для выбранного UART.
- `void RCC_UARTClkCmd (NT_UART_TypeDef *UARTx, FunctionalState State)`
Включение тактирования UART.
- `void RCC_SPIClkSel (NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)`
Настройка источника тактового сигнала для выбранного SPI.
- `void RCC_SPIClkDivConfig (NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)`
Настройка делителя тактового сигнала для выбранного SPI.
- `void RCC_SPIClkCmd (NT_SPI_TypeDef *SPIx, FunctionalState State)`
Включение тактирования SPI.
- `void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)`
Настройка делителя тактового сигнала для выбранного ADC.
- `void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)`
Включение тактирования ADC.
- `void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)`
Вывод из состояния сброса периферийных блоков.

6.126.1 Подробное описание

6.126.2 Функции

6.126.2.1 void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 779

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_ADC_CLK, RCC_ADCClk_0, RCC_ADCClk_1, RCC_ADCClk_10, RCC_ADCClk_2, RCC_ADCClk_3, RCC_ADCClk_4, RCC_ADCClk_5, RCC_ADCClk_6, RCC_ADCClk_7, RCC_ADCClk_8 и RCC_ADCClk_9.

6.126.2.2 void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 695

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_ADC_CLK, IS_RCC_CLK_DIV, RCC_ADCClk_0, RCC_ADCClk_1, RCC_ADCClk_10, RCC_ADCClk_2, RCC_ADCClk_3, RCC_ADCClk_4, RCC_ADCClk_5, RCC_ADCClk_6, RCC_ADCClk_7, RCC_ADCClk_8 и RCC_ADCClk_9.

6.126.2.3 void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)

Включение тактирования выбранного блока периферии.

Внимание

Блоки UART , SPI, ADC, USB управляются отдельно.

- [Тактирование UART](#)
- [Тактирование SPI](#)

- [Тактирование ADC](#)
- [Тактирование USB](#)

Аргументы

RCC_Periph↔ Clk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_↔TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 342

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_RCC_PERIPH_CLK`.

6.126.2.4 `void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)`

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_Periph↔ Rst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_↔_PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 869

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_PERIPH_RST` и `RCC_PeriphRst_↔ETH`.

Используется в `CAP_DeInit()` и `UART_DeInit()`.

6.126.2.5 `OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)`

Автоматическая конфигурация PLL для получения желаемой системной частоты.

С учетом данных об источнике частоты для PLL, а также о значении желаемой частоты, вычисляются все необходимые коэффициенты.

Внимание

Если `Freq < 50 МГц`, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

RCC_PLLRef	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из RCC_PLLRef_TypeDef .
------------	---

SysFreq	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.
---------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 142

Перекрестные ссылки EXT_OSC_VALUE, IS_RCC_PLL_REF, IS_RCC_SYS_FREQ, RCC_CLK_PLL_STABLE_TIMEOUT, RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit(), RCC_PLLInit_TypeDef::RCC_PLLNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLNO_Div2, RCC_PLLNO_Div4, RCC_PLLInit_TypeDef::RCC_PLLNR, RCC_PLLInit_TypeDef::RCC_PLLRef, RCC_PLLRef_ETH_25MHz, RCC_PLLRef_USB_60MHz, RCC_PLLRef_USB_CLK, RCC_PLLRef_XI_OSC, RCC_SysClk_PLL, RCC_SysClk_PLLDIV и RCC_SysClkSel().

6.126.2.6 void RCC_PLLDeInit ()

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 298

Перекрестные ссылки RCC_PLL_CTRL_Reset_Value, RCC_PLL_NF_Reset_Value, RCC_PLLNR_Reset_Value и RCC_PLL_OD_Reset_Value.

6.126.2.7 void RCC_PLLInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Инициализирует PLL согласно параметрам структуры RCC_PLLInit_Struct.

Значение выходной частоты PLL вычисляется с использованием значений опорного NR и выходного NO делителей, а также делителя обратной связи NF по формуле:

$$F_{OUT} = (F_{IN} \times NF) / (NO \times NR),$$

где F_{IN} – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- $3,2 \text{ МГц} < F_{IN} < 150 \text{ МГц}$,
- $800 \text{ КГц} < F_{REF} < 8 \text{ МГц}$,
- $200 \text{ МГц} < F_{VCO} < 500 \text{ МГц}$,

где частота фазового детектора F_{REF} вычисляется по формуле:

$$F_{REF} = F_{IN} / (2 \times NR),$$

а частота FVCO вычисляется по формуле:

$$FVCO = FIN \times (NF / NR)$$

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 262

Перекрестные ссылки IS_RCC_PLL_NF, IS_RCC_PLL_NO, IS_RCC_PLL_NR, IS_RCC_PL↔L_REF, IS_RCC_PLLDIV, RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit_TypeDef::RC↔C_PLLNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLInit_TypeDef::RCC_PLLNR и RC↔C_PLLInit_TypeDef::RCC_PLLRef.

Используется в RCC_PLLAutoConfig().

6.126.2.8 void RCC_PLLPowerDownCmd (FunctionalState State)

Управление режимом PowerDown PLL.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 313

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.126.2.9 void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 284

Перекрестные ссылки RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit_TypeDef::RCC_PL↔LNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLNO_Disable, RCC_PLLInit_TypeDef::R↔CC_PLLNR, RCC_PLLInit_TypeDef::RCC_PLLRef и RCC_PLLRef_XI_OSC.

6.126.2.10 void RCC_SPIClkCmd (NT_SPI_TypeDef * SPIx, FunctionalState State)

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 648

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_SPI_ALL_PERIPH`.

6.126.2.11 `void RCC_SPIClkDivConfig (NT_SPI_TypeDef * SPIx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 610

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_SPI_ALL_PERIPH`.

6.126.2.12 `void RCC_SPIClkSel (NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)`

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 569

Перекрестные ссылки `IS_RCC_SPI_CLK` и `IS_SPI_ALL_PERIPH`.

6.126.2.13 `void RCC_SysClkDiv2Out (FunctionalState State)`

Включение генерации тактового сигнала с частотой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState: <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 106

Перекрестные ссылки GPIO_InitTypeDef::GPIO_AltFunc, GPIO_AltFunc_2, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_Out, GPIO_Init(), GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_0, GPIO_StructInit() и IS_FUNCTIONAL_STATE.

6.126.2.14 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_TypeDef .
------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 365

Перекрестные ссылки IS_RCC_SYS_CLK и RCC_WaitClkChange().

Используется в RCC_PLLAutoConfig().

6.126.2.15 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле niietcm4_rcc.c строка 394

6.126.2.16 void RCC_UARTClkCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 526

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.126.2.17 `void RCC_UARTClkDivConfig (NT_UART_TypeDef * UARTx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 488

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_UART_ALL_PERIPH`.

6.126.2.18 `void RCC_UARTClkSel (NT_UART_TypeDef * UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)`

Настройка источника тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
RCC_UARTClk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 448

Перекрестные ссылки `IS_RCC_UART_CLK` и `IS_UART_ALL_PERIPH`.

6.126.2.19 `void RCC_USBClkCmd (FunctionalState State)`

Включение тактирования USB.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 425

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

```
6.126.2.20 void RCC_USBClkConfig ( RCC_USBClk_TypeDef RCC_USBClk,
                                RCC_USBFreq_TypeDef RCC_USBFreq )
```

Настройка источника тактового сигнала для USB.

Аргументы

<code>RCC_USBClk</code>	Выбор источника тактирования. Параметр принимает любое значение из RCC_CUSBClk_TypeDef .
<code>RCC_USB_Clock_Freq</code>	Выбор фиксированной частоты на входе <code>CLK_USB</code> . Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 408

Перекрестные ссылки `IS_RCC_USB_CLK` и `IS_RCC_USB_FREQ`.

```
6.126.2.21 uint32_t RCC_WaitClkChange ( RCC_SysClk_TypeDef RCC_SysClk )
```

Процедура ожидания смены источника тактового сигнала

Возвращаемые значения

<code>timeout</code>	Значение остатка времени после ожидания.
----------------------	--

См. определение в файле `niietcm4_rcc.c` строка 77

Перекрестные ссылки `RCC_CLK_CHANGE_TIMEOUT`.

Используется в `RCC_SysClkSel()`.

6.127 RTC

Драйвер для модуля часов реального времени.

Группы

- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.127.1 Подробное описание

Драйвер для модуля часов реального времени.

6.128 Приватные данные

Группы

- [Приватные функции](#)

6.128.1 Подробное описание

6.129 Приватные функции

Функции

- uint32_t bcd2hex (uint32_t a)
- uint32_t hex2bcd (uint32_t x)
- void RTC_ShadowUpd ([FunctionalState](#) State)
- void RTC_GetTime ([RTC_Format_TypeDef](#) RTC_Format, [RTC_Time_TypeDef](#) *RTC_Time)
- void RTC_GetDate ([RTC_Format_TypeDef](#) RTC_Format, [RTC_Date_TypeDef](#) *RTC_Date)
- void RTC_SetTime ([RTC_Format_TypeDef](#) RTC_Format, [RTC_Time_TypeDef](#) *RTC_Time)
- void RTC_SetDate ([RTC_Format_TypeDef](#) RTC_Format, [RTC_Date_TypeDef](#) *RTC_Date)

6.129.1 Подробное описание

6.130 TIMER

Драйвер для таймеров

Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

6.130.1 Подробное описание

Драйвер для таймеров

6.131 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.131.1 Подробное описание

6.132 Приватные константы

6.133 Приватные функции

Функции

- void **TIMER_Cmd** (NT_TIMER_TypeDef *TIMERx, [FunctionalState](#) State)
Разрешение работы выбранного таймера.
- void **TIMER_PeriodConfig** (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)
Настройка периода опустошения выбранного таймера.
- void **TIMER_FreqConfig** (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)
Настройка частоты опустошения выбранного таймера.
- void **TIMER_SetReload** (NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)
Установка значения перезагрузки.
- uint32_t **TIMER_GetReload** (NT_TIMER_TypeDef *TIMERx)
Получение текущего значения перезагрузки.
- void **TIMER_SetCounter** (NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)
Установка значения счетчика.
- uint32_t **TIMER_GetCounter** (NT_TIMER_TypeDef *TIMERx)
Получение текущего значения счетчика.
- void **TIMER_ExtInputConfig** (NT_TIMER_TypeDef *TIMERx, [TIMER_ExtInput_TypeDef](#) [TIMER_ExtInput](#))
Выбор режима работы входа внешнего тактирования.
- void **TIMER_ITCmd** (NT_TIMER_TypeDef *TIMERx, [FunctionalState](#) State)
Разрешение работы прерывания выбранного таймера.
- [FlagStatus](#) **TIMER_ITStatus** (NT_TIMER_TypeDef *TIMERx)
Чтение статуса прерывания выбранного таймера.
- void **TIMER_ITStatusClear** (NT_TIMER_TypeDef *TIMERx)
Очищение статусного бита прерывания выбранного таймера.

6.133.1 Подробное описание

6.133.2 Функции

6.133.2.1 void **TIMER_Cmd** (NT_TIMER_TypeDef * TIMERx, [FunctionalState](#) State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 65

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#) и [IS_TIMER_ALL_PERIPH](#).

6.133.2.2 void **TIMER_ExtInputConfig** (NT_TIMER_TypeDef * TIMERx, [TIMER_ExtInput_TypeDef](#) [TIMER_ExtInput](#))

Выбор режима работы входа внешнего тактирования.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TIMER_Ext↔ Input	Выбор режима работы. Параметр принимает любое значение из TIMER_Ext↔ Input_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 171

Перекрестные ссылки IS_TIMER_ALL_PERIPH, IS_TIMER_EXT_INPUT, TIMER_ExtInput↔_CountClk и TIMER_ExtInput_CountEn.

6.133.2.3 void TIMER_FreqConfig (NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)

Настройка частоты опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемого периода опустошения таймера [TIMER_PeriodConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerFreq	Частота опустошения таймера в Гц.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 102

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.4 uint32_t TIMER_GetCounter (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_timer.c строка 156

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.5 uint32_t TIMER_GetReload (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_timer.c строка 129

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.6 void TIMER_ITCmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

6.133.2.7 FlagStatus TIMER_ITStatus (NT_TIMER_TypeDef * TIMERx)

Чтение статуса прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_timer.c строка 214

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.8 void TIMER_ITStatusClear (NT_TIMER_TypeDef * TIMERx)

Очищение статусного бита прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 238

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.9 void TIMER_PeriodConfig (NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)

Настройка периода опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемой частоты опустошения таймера [TIMER_FreqConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerPeriod	Период опустошения таймера в мкс.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 84

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.10 void TIMER_SetCounter (NT_TIMER_TypeDef * TIMERx, uint32_t CounterVal)

Установка значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
CounterVal	Значение счетчика.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 143

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.133.2.11 void TIMER_SetReload (NT_TIMER_TypeDef * TIMERx, uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
ReloadVal	Значение перезагрузки.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 116

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.134 UART

Драйвер для приемопередатчиков UART.

Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

6.134.1 Подробное описание

Драйвер для приемопередатчиков UART.

Внимание

Драйвер позволяет управлять только внутренними настройками модулей UART. Соответствующие порты ввода-вывода, а также системное тактирование и сброс блоков необходимо настраивать отдельно.

GPIO : [Инициализация и деинициализация](#).

RCC : [Тактирование UART](#), [Управление сбросом](#).

6.135 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.135.1 Подробное описание

6.136 Приватные константы

6.137 Приватные функции

Функции

- void **UART_Cmd** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Разрешение работы выбранного UART.
- void **UART_BaudRateDivConfig** (NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t ←
t FracDiv)
Ручная настройка делителя для реализации необходимой скорости передачи.
- void **UART_Break** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Включение разрыва линии.
- void **UART_DeInit** (NT_UART_TypeDef *UARTx)
Устанавливает все регистры UART значениями по умолчанию.
- **OperationStatus** **UART_Init** (NT_UART_TypeDef *UARTx, **UART_Init_TypeDef** *UART ←
_InitStruct)
Инициализирует UARTx согласно параметрам структуры UART_InitStruct.
- void **UART_StructInit** (**UART_Init_TypeDef** *UART_InitStruct)
Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.
- void **UART_SendData** (NT_UART_TypeDef *UARTx, uint32_t Data)
Передача слова данных.
- uint32_t **UART_RecieveData** (NT_UART_TypeDef *UARTx)
Прием слова данных.
- **FlagStatus** **UART_FlagStatus** (NT_UART_TypeDef *UARTx, **UART_Flag_Typedef** UART ←
_Flag)
Запрос состояния выбранного флага.
- **FlagStatus** **UART_ErrorStatus** (NT_UART_TypeDef *UARTx, **UART_Error_Typedef** UAR ←
T_Error)
Запрос состояния выбранного флага ошибки.
- void **UART_ErrorStatusClear** (NT_UART_TypeDef *UARTx)
Очистка флагов ошибки.
- void **UART_ModemConfig** (NT_UART_TypeDef *UARTx, **UART_ModemInit_TypeDef** *U ←
ART_ModemInitStruct)
Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInit ←
Struct.
- void **UART_ModemStructInit** (**UART_ModemInit_TypeDef** *UART_ModemInitStruct)
Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.
- void **UART_ITFIFOLevelConfig** (NT_UART_TypeDef *UARTx, **UART_Dir_Typedef** UAR ←
T_Dir, **UART_FIFOLevel_TypeDef** UART_FIFOLevel)
Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генери-
роваться прерывание.
- void **UART_ITCmd** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_IT ←
Source, **FunctionalState** State)
Маскирование выбранных прерываний.
- **FlagStatus** **UART_ITRawStatus** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** ←
UART_ITSource)
Запрос немаскированного состояния прерывания.
- **FlagStatus** **UART_ITMaskedStatus** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** ←
UART_ITSource)
Запрос маскированного состояния прерывания.
- void **UART_ITStatusClear** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UAR ←
T_ITSource)
Сброс флагов состояния выбранных прерываний.
- void **UART_DMABlkOnErrCmd** (NT_UART_TypeDef *UARTx, **FunctionalState** State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

- void [UART_DMAMCmd](#) (NT_UART_TypeDef *UARTx, [UART_Dir_Typedef](#) UART_Dir, [FunctionalState](#) State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

6.137.1 Подробное описание

6.137.2 Функции

- 6.137.2.1 void [UART_BaudRateDivConfig](#) (NT_UART_TypeDef * UARTx, uint32_t IntDiv, uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

- 6.137.2.2 void [UART_Break](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

- 6.137.2.3 void [UART_Cmd](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 69

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.137.2.4 `void UART_DeInit (NT_UART_TypeDef * UARTx)`

Устанавливает все регистры UART значениями по умолчанию.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
-------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 120

Перекрестные ссылки `IS_UART_ALL_PERIPH`, `RCC_PeriphRst_UART0`, `RCC_PeriphRst_UART1`, `RCC_PeriphRst_UART2`, `RCC_PeriphRst_UART3` и `RCC_PeriphRstCmd()`.

6.137.2.5 `void UART_DMABlkOnErrCmd (NT_UART_TypeDef * UARTx, FunctionalState State)`

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 502

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.137.2.6 `void UART_DMACmd (NT_UART_TypeDef * UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)`

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 520

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_UART_ALL_PERIPH`, `IS_UART_DIR` и `UART_Dir_Rx`.

6.137.2.7 FlagStatus UART_ErrorStatus (NT_UART_TypeDef * UARTx, UART_Error_TypeDef UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

6.137.2.8 void UART_ErrorStatusClear (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.137.2.9 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_TypeDef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

6.137.2.10 OperationStatus UART_Init (NT_UART_TypeDef * UARTx, UART_Init_TypeDef * UART_InitStruct)

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_Init↔ Struct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле niietcm4_uart.c строка 159

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DATA_WIDTH, IS_UART_FIFO_LEVEL, IS_UART_PARITY_BIT, IS_UART_STOP_BIT, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_Init_TypeDef::UART_FIFOEn, UART_Init_TypeDef::UART_FIFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Even, UART_ParityBit_High, UART_ParityBit_Low, UART_ParityBit_Odd, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit и UART_Init_TypeDef::UART_TxEn.

```
6.137.2.11 void UART_ITCmd ( NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource, FunctionalState State )
```

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 410

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

```
6.137.2.12 void UART_ITFIFOLevelConfig ( NT_UART_TypeDef * UARTx, UART_Dir_TypeDef
UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel )
```

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор между буфером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFO↔ OLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и UART_Dir_Rx.

6.137.2.13 `FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)`

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле `niietcm4_uart.c` строка 459

Перекрестные ссылки `IS_UART_ALL_PERIPH` и `IS_UART_GET_IT_SOURCE`.

6.137.2.14 `FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)`

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле `niietcm4_uart.c` строка 433

Перекрестные ссылки `IS_UART_ALL_PERIPH` и `IS_UART_GET_IT_SOURCE`.

6.137.2.15 `void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)`

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 485

Перекрестные ссылки `IS_UART_ALL_PERIPH` и `IS_UART_IT_SOURCE`.

6.137.2.16 `void UART_ModemConfig (NT_UART_TypeDef * UARTx,
UART_ModemInit_TypeDef * UART_ModemInitStruct)`

Инициализирует модемный режим UART согласно параметрам структуры `UART_ModemInit←
Struct`.

Аргументы

UART_↔ ModemInit_↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
---------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_Modem↔Init_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit↔_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

```
6.137.2.17 void UART_ModemStructInit ( UART_ModemInit_TypeDef * UART_ModemInitStruct )
```

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_↔ ModemInit_↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
---------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef↔::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::↔UART_RTSEn.

```
6.137.2.18 uint32_t UART_RecieveData ( NT_UART_TypeDef * UARTx )
```

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

```
6.137.2.19 void UART_SendData ( NT_UART_TypeDef * UARTx, uint32_t Data )
```

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 250

Перекрестные ссылки `IS_UART_ALL_PERIPH` и `IS_UART_DATA`.

6.137.2.20 `void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)`

Заполнение каждого члена структуры `UART_InitStruct` значениями по умолчанию.

Аргументы

<code>UART_InitStruct</code>	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 229

Перекрестные ссылки `EXT_OSC_VALUE`, `UART_Init_TypeDef::UART_BaudRate`, `UART_Init_TypeDef::UART_ClkFreq`, `UART_Init_TypeDef::UART_DataWidth`, `UART_DataWidth_8`, `UART_Init_TypeDef::UART_FIFOEn`, `UART_FIFOLevel_1_2`, `UART_Init_TypeDef::UART_FIFOLevelRx`, `UART_Init_TypeDef::UART_FIFOLevelTx`, `UART_Init_TypeDef::UART_ParityBit`, `UART_ParityBit_Disable`, `UART_Init_TypeDef::UART_RxEn`, `UART_Init_TypeDef::UART_StopBit`, `UART_StopBit_1` и `UART_Init_TypeDef::UART_TxEn`.

6.138 USERFLASH

Драйвер для пользовательской флеш-памяти.

Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.138.1 Подробное описание

Драйвер для пользовательской флеш-памяти.

Внимание

Перед использованием какой либо функции этого драйвера, следует обязательно произвести инициализацию контроллера памяти - [USERFLASH_Init\(\)](#).

6.139 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.139.1 Подробное описание

6.140 Приватные константы

6.141 Приватные функции

Функции

- void **USERFLASH_Init** (uint32_t SysClkFreq)
Инициализирует тайминги доступа для контроллера пользовательской флеш.
- **USERFLASH_Status_TypeDef USERFLASH_OperationStatus** ()
Статус работы контроллера пользовательской флэш.
- void **USERFLASH_OperationStatusClear** ()
Очищает статус работы контроллера пользовательской флэш.
- void **USERFLASH_FullErase** ()
Полная очистка основной области пользовательской флеш.
- uint32_t **USERFLASH_Read** (uint32_t Address)
Чтение байта из основной области пользовательской флеш.
- void **USERFLASH_Write** (uint32_t Address, uint32_t Data)
Запись байта в основную область пользовательской флеш по указанному адресу.
- void **USERFLASH_PageErase** (uint32_t PageNum)
Стирание указанной страницы основной области пользовательской флеш.
- uint32_t **USERFLASH_Info_Read** (uint32_t Address)
Чтение байта из информационной области пользовательской флеш.
- void **USERFLASH_Info_Write** (uint32_t Address, uint32_t Data)
Запись байта в информационную область пользовательской флеш по указанному адресу.
- void **USERFLASH_Info_PageErase** (uint32_t PageNum)
Стирание указанной страницы информационной области пользовательской флеш.
- void **USERFLASH_ITCmd** (FunctionalState State)
Включение прерывания по завершению чтения/записи/стирания.

6.141.1 Подробное описание

6.141.2 Функции

6.141.2.1 void USERFLASH_FullErase ()

Полная очистка основной области пользовательской флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 103

Перекрестные ссылки **USERFLASH_MAGIC_KEY**, **USERFLASH_OperationStatus()** и **USERFLASH_Status_None**.

6.141.2.2 void USERFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области пользовательской флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 219

Перекрестные ссылки IS_USERFLASH_INFO_PAGE_NUM, USERFLASH_MAGIC_KEY и USERFLASH_PAGE_SIZE_BYTES.

6.141.2.3 uint32_t USERFLASH_Info_Read (uint32_t Address)

Чтение байта из информационной области пользовательской флеш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 175

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

6.141.2.4 void USERFLASH_Info_Write (uint32_t Address, uint32_t Data)

Запись байта в информационную область пользовательской флеш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 206

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.141.2.5 void USERFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера пользовательской флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
------------	---------------------------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 66

6.141.2.6 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.141.2.7 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флэш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из USERFLASH_Status_TypeDef .
--------	---

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

6.141.2.8 void USERFLASH_OperationStatusClear ()

Очищает статус работы контроллера пользовательской флэш.

Возвращаемые значения

Нет.

См. определение в файле niietcm4_userflash.c строка 93

Используется в USERFLASH_Info_Read() и USERFLASH_Read().

6.141.2.9 void USERFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области пользовательской флэш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 161

Перекрестные ссылки IS_USERFLASH_PAGE_NUM, USERFLASH_MAGIC_KEY и USERFLASH_PAGE_SIZE_BYTES.

6.141.2.10 uint32_t USERFLASH_Read (uint32_t Address)

Чтение байта из основной области пользовательской флэш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 116

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

6.141.2.11 void USERFLASH_Write (uint32_t Address, uint32_t Data)

Запись байта в основную область пользовательской флеш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 148

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.142 WATCHDOG

Драйвер для сторожевого таймера

Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

6.142.1 Подробное описание

Драйвер для сторожевого таймера

6.143 Приватные данные

Группы

- [Приватные константы](#)
- [Приватные функции](#)

6.143.1 Подробное описание

6.144 Приватные константы

Макросы

- `#define WATCHDOG_Lock_Value ((uint32_t)0xDEADC0DE)`
- `#define WATCHDOG_Unlock_Value ((uint32_t)0x1ACCE551)`

6.144.1 Подробное описание

6.144.2 Макросы

6.144.2.1 `#define WATCHDOG_Lock_Value ((uint32_t)0xDEADC0DE)`

Любое значение для блокировки записи в регистры таймера

См. определение в файле `niietcm4_watchdog.c` строка 50

Используется в `WATCHDOG_LockCmd()`.

6.144.2.2 `#define WATCHDOG_Unlock_Value ((uint32_t)0x1ACCE551)`

Значение для разблокировки записи в регистры таймера

См. определение в файле `niietcm4_watchdog.c` строка 51

Используется в `WATCHDOG_LockCmd()`.

6.145 Приватные функции

Функции

- void [WATCHDOG_Cmd](#) ([FunctionalState](#) State)
Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.
- void [WATCHDOG_SetReload](#) (uint32_t ReloadVal)
Установка значения перезагрузки.
- uint32_t [WATCHDOG_GetReload](#) ()
Получение текущего значения перезагрузки.
- uint32_t [WATCHDOG_GetCounter](#) ()
Получение текущего значения счетчика.
- void [WATCHDOG_RstCmd](#) ([FunctionalState](#) State)
Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.
- void [WATCHDOG_LockCmd](#) ([FunctionalState](#) State)
Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.
- [FlagStatus](#) [WATCHDOG_ITRawStatus](#) ()
Чтение немаскированного флага прерывания сторожевого таймера.
- [FlagStatus](#) [WATCHDOG_ITMaskedStatus](#) ()
Чтение маскированного флага прерывания сторожевого таймера.
- void [WATCHDOG_ITStatusClear](#) ()
Очищение статусного бита прерывания сторожевого таймера.

6.145.1 Подробное описание

6.145.2 Функции

6.145.2.1 void [WATCHDOG_Cmd](#) ([FunctionalState](#) State)

Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 69

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.145.2.2 uint32_t [WATCHDOG_GetCounter](#) ()

Получение текущего значения счетчика.

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_watchdog.c строка 105

6.145.2.3 uint32_t [WATCHDOG_GetReload](#) ()

Получение текущего значения перезагрузки.

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_watchdog.c строка 95

6.145.2.4 FlagStatus WATCHDOG_ITMaskedStatus ()

Чтение маскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 174

6.145.2.5 FlagStatus WATCHDOG_ITRawStatus ()

Чтение немаскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 153

6.145.2.6 void WATCHDOG_ITStatusClear ()

Очищение статусного бита прерывания сторожевого таймера.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_watchdog.c строка 195

6.145.2.7 void WATCHDOG_LockCmd (FunctionalState State)

Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_watchdog.c строка 134

Перекрестные ссылки IS_FUNCTIONAL_STATE, WATCHDOG_Lock_Value и WATCHDOG_Unlock_Value.

6.145.2.8 void WATCHDOG_RstCmd (FunctionalState State)

Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 119

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.145.2.9 void WATCHDOG_SetReload (uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

ReloadVal	Значение перезагрузки. Параметр принимает любое значение из диапазона 0x1 - 0xFFFFFFFF.
-----------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 83

Перекрестные ссылки IS_WATCHDOG_RELOAD.

6.146 Драйвер периферии

Группы

- [Настройка драйвера](#)
- [Макросы](#)
- [Типы](#)
- [Периферия](#)

6.146.1 Подробное описание

6.147 Настройка драйвера

Макросы

- `#define EXT_OSC_VALUE ((uint32_t)12000000)`
Определение частоты используемого внешнего тактового генератора.
- `#define INT_OSC_VALUE ((uint32_t)8000000)`
Определение частоты частоты внутреннего тактового генератора.

6.147.1 Подробное описание

6.147.2 Макросы

6.147.2.1 `#define EXT_OSC_VALUE ((uint32_t)12000000)`

Определение частоты используемого внешнего тактового генератора.

Совет: Чтобы избежать необходимости каждый раз изменять этот файл, можно передать определение устройства компилятору через ключ.

Например, для GCC ARM это выглядит так:

`-DEXT_OSC_VALUE=16000000`

Частота внешнего тактового генератора [Гц].

См. определение в файле `niietcm4.h` строка 73

Используется в `RCC_PLLAutoConfig()` и `UART_StructInit()`.

6.147.2.2 `#define INT_OSC_VALUE ((uint32_t)8000000)`

Определение частоты частоты внутреннего тактового генератора.

Конфигурируется автоматически в зависимости от выбранного целевого устройства. Частота внутреннего тактового генератора [Гц].

См. определение в файле `niietcm4.h` строка 88

6.148 Макросы

Макросы

- `#define SET_BIT(REG, BIT) ((REG) |= (BIT))`
Установить бит в регистре.
- `#define CLEAR_BIT(REG, BIT) ((REG) &= ~(BIT))`
Сбросить бит в регистре.
- `#define READ_BIT(REG, BIT) ((REG) & (BIT))`
Прочитать бит из регистра.
- `#define CLEAR_REG(REG) ((REG) = (0x0))`
Обнулить значение регистра.
- `#define WRITE_REG(REG, VAL) ((REG) = (VAL))`
Записать значение в регистр.
- `#define READ_REG(REG) ((REG))`
Прочитать значение из регистра.
- `#define MODIFY_REG(REG, CLEARMASK, SETMASK) WRITE_REG((REG), (((READ_REG(REG) & (~CLEARMASK))) | (SETMASK)))`
Изменить значение регистра по маске.

6.148.1 Подробное описание

6.149 Типы

Макросы

- `#define IS_FUNCTIONAL_STATE(STATE) (((STATE) == DISABLE) || ((STATE) == ENABLE))`
Макрос проверки аргументов типа `FunctionalState`.
- `#define IS_TIMER_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_TIMER_TypeDef`.
- `#define IS_GPIO_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_GPIO_TypeDef`.
- `#define IS_UART_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_UART_TypeDef`.
- `#define IS_SPI_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_SPI_TypeDef`.
- `#define IS_CAP_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_CAP_TypeDef`.

Перечисления

- `enum FunctionalState { DISABLE = 0, ENABLE = 1 }`
Описывает логическое состояние периферии. Используется для операций включения/выключения периферийных блоков.
- `enum OperationStatus { OK = 0, ERROR = 1 }`
Описывает коды возврата для функций при выполнении какой-либо операции.
- `enum FlagStatus { Flag_CLEAR = 0, Flag_SET = 1 }`
Описывает возможные состояния флага при запросе его статуса.

6.149.1 Подробное описание

6.149.2 Макросы

6.149.2.1 `#define IS_CAP_ALL_PERIPH(PERIPH)`

Макроопределение:

```
((PERIPH) == NT_CAP0) || \
    ((PERIPH) == NT_CAP1) || \
    ((PERIPH) == NT_CAP2) || \
    ((PERIPH) == NT_CAP3) || \
    ((PERIPH) == NT_CAP4) || \
    ((PERIPH) == NT_CAP5)
```

Макрос проверки аргументов типа `NT_CAP_TypeDef`.

См. определение в файле `niietcm4.h` строка 232

Используется в `CAP_Capture_Cmd()`, `CAP_Capture_GetCap0()`, `CAP_Capture_GetCap1()`, `CAP_Capture_GetCap2()`, `CAP_Capture_GetCap3()`, `CAP_Capture_Init()`, `CAP_Capture_SetCap0()`, `CAP_Capture_SetCap1()`, `CAP_Capture_SetCap2()`, `CAP_Capture_SetCap3()`, `CAP_DeInit()`, `CAP_GetShadowTimer()`, `CAP_GetTimer()`, `CAP_Init()`, `CAP_ITCmd()`, `CAP_ITForceCmd()`, `CAP_ITPendClear()`, `CAP_ITPendStatus()`, `CAP_ITStatus()`, `CAP_ITStatusClear()`, `CAP_PWM_GetCompare()`, `CAP_PWM_GetPeriod()`, `CAP_PWM_GetShadowCompare()`, `CAP_PWM_GetShadowPeriod()`, `CAP_PWM_Init()`, `CAP_PWM_SetCompare()`, `CAP_PWM_SetPeriod()`, `CAP_PWM_SetShadowCompare()`, `CAP_PWM_SetShadowPeriod()`, `CAP_SetShadowTimer()`, `CAP_SetTimer()`, `CAP_SwSync()`, `CAP_SyncCmd()` и `CAP_TimerCmd()`.

6.149.2.2 #define IS_GPIO_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_GPIOA) || \
  ((PERIPH) == NT_GPIOB) || \
  ((PERIPH) == NT_GPIOC) || \
  ((PERIPH) == NT_GPIOD) || \
  ((PERIPH) == NT_GPIOE) || \
  ((PERIPH) == NT_GPIOF) || \
  ((PERIPH) == NT_GPIOG) || \
  ((PERIPH) == NT_GPIOH))
```

Макрос проверки аргументов типа NT_GPIO_TypeDef.

См. определение в файле niietcm4.h строка 201

Используется в GPIO_ClearBits(), GPIO_DeInit(), GPIO_Init(), GPIO_ITCmd(), GPIO_ITConfig(), GPIO_ITStatusClear(), GPIO_QualCmd(), GPIO_QualConfig(), GPIO_Read(), GPIO_ReadBit(), GPIO_ReadMask(), GPIO_SetBits(), GPIO_SyncCmd(), GPIO_ToggleBits(), GPIO_Write(), GPIO_WriteBit() и GPIO_WriteMask().

6.149.2.3 #define IS_SPI_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_SPI0) || \
  ((PERIPH) == NT_SPI1) || \
  ((PERIPH) == NT_SPI2) || \
  ((PERIPH) == NT_SPI3))
```

Макрос проверки аргументов типа NT_SPI_TypeDef.

См. определение в файле niietcm4.h строка 223

Используется в RCC_SPIClkCmd(), RCC_SPIClkDivConfig() и RCC_SPIClkSel().

6.149.2.4 #define IS_TIMER_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_TIMER0) || \
  ((PERIPH) == NT_TIMER1) || \
  ((PERIPH) == NT_TIMER2))
```

Макрос проверки аргументов типа NT_TIMER_TypeDef.

См. определение в файле niietcm4.h строка 193

Используется в TIMER_Cmd(), TIMER_ExtInputConfig(), TIMER_FreqConfig(), TIMER_GetCounter(), TIMER_GetReload(), TIMER_ITCmd(), TIMER_ITStatus(), TIMER_ITStatusClear(), TIMER_PeriodConfig(), TIMER_SetCounter() и TIMER_SetReload().

6.149.2.5 #define IS_UART_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_UART0) || \
  ((PERIPH) == NT_UART1) || \
  ((PERIPH) == NT_UART2) || \
  ((PERIPH) == NT_UART3))
```

Макрос проверки аргументов типа NT_UART_TypeDef.

См. определение в файле niietcm4.h строка 214

Используется в `RCC_UARTClkCmd()`, `RCC_UARTClkDivConfig()`, `RCC_UARTClkSel()`, `UART_↵
_BaudRateDivConfig()`, `UART_Break()`, `UART_Cmd()`, `UART_DeInit()`, `UART_DMABlkOnErr↵
Cmd()`, `UART_DMACmd()`, `UART_ErrorStatus()`, `UART_ErrorStatusClear()`, `UART_FlagStatus()`,
`UART_Init()`, `UART_ITCmd()`, `UART_ITFIFOLevelConfig()`, `UART_ITMaskedStatus()`, `UART_↵
ITRawStatus()`, `UART_ITStatusClear()`, `UART_ModemConfig()`, `UART_RcieveData()` и `UART_↵
SendData()`.

6.150 Периферия

Группы

- [ADC](#)
Драйвер для модулей АЦП, связанных с ними секвенсоров, а также цифровых компараторов.
- [BOOTFLASH](#)
Драйвер для загрузочной флеш-памяти.
- [CAP](#)
Драйвер для блоков захвата
- [DMA](#)
Драйвер для работы с контроллером прямого доступа памяти.
- [EXTMEM](#)
Драйвер для интерфейса внешней памяти.
- [GPIO](#)
Драйвер для управления портами ввода-вывода.
- [RCC](#)
Драйвер для работы с тактированием и сбросом периферийных блоков.
- [RTC](#)
Драйвер для модуля часов реального времени.
- [TIMER](#)
Драйвер для таймеров
- [UART](#)
Драйвер для приемопередатчиков UART.
- [USERFLASH](#)
Драйвер для пользовательской флеш-памяти.
- [WATCHDOG](#)
Драйвер для сторожевого таймера

6.150.1 Подробное описание

Глава 7

Структуры данных

7.1 Структура `_CHANNEL_CFG_bits`

Битовый доступ к регистру `CHANNEL_CFG` в `DMA_Channel_TypeDef`.

```
#include <niietcm4_dma.h>
```

Поля данных

- `uint32_t CYCLE_CTRL:3`
- `uint32_t NEXT_USEBURST:1`
- `uint32_t N_MINUS_1:10`
- `uint32_t R_POWER:4`
- `uint32_t SRC_PROT_PRIVILEGED:1`
- `uint32_t SRC_PROT_BUFFERABLE:1`
- `uint32_t SRC_PROT_CACHEABLE:1`
- `uint32_t DST_PROT_PRIVILEGED:1`
- `uint32_t DST_PROT_BUFFERABLE:1`
- `uint32_t DST_PROT_CACHEABLE:1`
- `uint32_t SRC_SIZE:2`
- `uint32_t SRC_INC:2`
- `uint32_t DST_SIZE:2`
- `uint32_t DST_INC:2`

7.1.1 Подробное описание

Битовый доступ к регистру `CHANNEL_CFG` в `DMA_Channel_TypeDef`.

См. определение в файле `niietcm4_dma.h` строка 207

7.1.2 Поля

7.1.2.1 `uint32_t _CHANNEL_CFG_bits::CYCLE_CTRL`

Выбор режима работы DMA

См. определение в файле `niietcm4_dma.h` строка 208

Используется в `DMA_ChannelInit()`.

7.1.2.2 uint32_t _CHANNEL_CFG_bits::DST_INC

Шаг инкремента адреса приемника при записи

См. определение в файле niietcm4_dma.h строка 221

Используется в DMA_ChannelInit().

7.1.2.3 uint32_t _CHANNEL_CFG_bits::DST_PROT_BUFFERABLE

Защита шины при записи в приемник: буферизация

См. определение в файле niietcm4_dma.h строка 216

Используется в DMA_ChannelInit().

7.1.2.4 uint32_t _CHANNEL_CFG_bits::DST_PROT_CACHEABLE

Защита шины при записи в приемник: кэширование

См. определение в файле niietcm4_dma.h строка 217

Используется в DMA_ChannelInit().

7.1.2.5 uint32_t _CHANNEL_CFG_bits::DST_PROT_PRIVILEGED

Защита шины при записи в приемник: привелегированный доступ

См. определение в файле niietcm4_dma.h строка 215

Используется в DMA_ChannelInit().

7.1.2.6 uint32_t _CHANNEL_CFG_bits::DST_SIZE

Разрядность данных приемника

См. определение в файле niietcm4_dma.h строка 220

Используется в DMA_ChannelInit().

7.1.2.7 uint32_t _CHANNEL_CFG_bits::N_MINUS_1

Общее количество передач N (в поле пишется значение N-1)

См. определение в файле niietcm4_dma.h строка 210

Используется в DMA_ChannelInit().

7.1.2.8 uint32_t _CHANNEL_CFG_bits::NEXT_USEBURST

Контроль установки соответствующего каналу бита в регистре NT_DMA->CHNL_USEBURST_↔ SET

См. определение в файле niietcm4_dma.h строка 209

Используется в DMA_ChannelInit().

7.1.2.9 uint32_t _CHANNEL_CFG_bits::R_POWER

Выбор количества передач до выполнения переарбитрации

См. определение в файле niietcm4_dma.h строка 211

Используется в DMA_ChannelInit().

7.1.2.10 uint32_t _CHANNEL_CFG_bits::SRC_INC

Шаг инкремента адреса источника при чтении

См. определение в файле niietcm4_dma.h строка 219

Используется в DMA_ChannelInit().

7.1.2.11 uint32_t _CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE

Защита шины при чтении из источника: буферизация

См. определение в файле niietcm4_dma.h строка 213

Используется в DMA_ChannelInit().

7.1.2.12 uint32_t _CHANNEL_CFG_bits::SRC_PROT_CACHEABLE

Защита шины при чтении из источника: кэширование

См. определение в файле niietcm4_dma.h строка 214

Используется в DMA_ChannelInit().

7.1.2.13 uint32_t _CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED

Защита шины при чтении из источника: привелегированный доступ

См. определение в файле niietcm4_dma.h строка 212

Используется в DMA_ChannelInit().

7.1.2.14 uint32_t _CHANNEL_CFG_bits::SRC_SIZE

Разрядность данных источника

См. определение в файле niietcm4_dma.h строка 218

Используется в DMA_ChannelInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.2 Структура ADC_DC_Init_TypeDef

Структура инициализации цифровых компараторов.

```
#include <niietcm4_adc.h>
```

Поля данных

- [uint32_t ADC_DC_ThresholdLow](#)
- [uint32_t ADC_DC_ThresholdHigh](#)
- [ADC_DC_Channel_TypeDef ADC_DC_Channel](#)
- [ADC_DC_Mode_TypeDef ADC_DC_Mode](#)
- [ADC_DC_Condition_TypeDef ADC_DC_Condition](#)

7.2.1 Подробное описание

Структура инициализации цифровых компараторов.

Внимание

- Условие срабатывания по попаданию в диапазон не работает с гистерезисными режимами работы.
- Должно всегда выполняться условие `ADC_DC_ThresholdLow <= ADC_DC_ThresholdHigh`.

См. определение в файле `niietcm4_adc.h` строка 599

7.2.2 Поля

7.2.2.1 `ADC_DC_Channel_TypeDef ADC_DC_Init_TypeDef::ADC_DC_Channel`

Выбирает канал, результат измерения которого будет передан на компаратор. Параметр может принимать любое значение из [ADC_DC_Channel_TypeDef](#).

См. определение в файле `niietcm4_adc.h` строка 605

Используется в `ADC_DC_Init()` и `ADC_DC_StructInit()`.

7.2.2.2 `ADC_DC_Condition_TypeDef ADC_DC_Init_TypeDef::ADC_DC_Condition`

Выбирает условие срабатывания компаратора. Параметр может принимать любое значение из [ADC_DC_Condition_TypeDef](#).

См. определение в файле `niietcm4_adc.h` строка 609

Используется в `ADC_DC_Init()` и `ADC_DC_StructInit()`.

7.2.2.3 `ADC_DC_Mode_TypeDef ADC_DC_Init_TypeDef::ADC_DC_Mode`

Выбирает режим срабатывания компаратора. Параметр может принимать любое значение из [ADC_DC_Mode_TypeDef](#).

См. определение в файле `niietcm4_adc.h` строка 607

Используется в `ADC_DC_Init()` и `ADC_DC_StructInit()`.

7.2.2.4 `uint32_t ADC_DC_Init_TypeDef::ADC_DC_ThresholdHigh`

Верхний порог срабатывания компаратора. Параметр может принимать любое значение из диапазона 0 - 4095.

См. определение в файле `niietcm4_adc.h` строка 603

Используется в `ADC_DC_Init()` и `ADC_DC_StructInit()`.

7.2.2.5 `uint32_t ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow`

Нижний порог срабатывания компаратора. Параметр может принимать любое значение из диапазона 0 - 4095.

См. определение в файле `niietcm4_adc.h` строка 601

Используется в `ADC_DC_Init()` и `ADC_DC_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_adc.h](#)

7.3 Структура ADC_Init_TypeDef

Структура инициализации модулей АЦП

```
#include <niietcm4_adc.h>
```

Поля данных

- [ADC_Resolution_TypeDef ADC_Resolution](#)
- [ADC_Measure_TypeDef ADC_Measure_A](#)
- [ADC_Measure_TypeDef ADC_Measure_B](#)
- `uint32_t ADC_Phase`
- [ADC_Average_TypeDef ADC_Average](#)
- [ADC_Mode_TypeDef ADC_Mode](#)

7.3.1 Подробное описание

Структура инициализации модулей АЦП

Внимание

Нельзя устанавливать оба канала в дифференциальный режим (параметры [ADC_Measure_A](#) и [ADC_Measure_B](#)).

См. определение в файле `niietcm4_adc.h` строка 576

7.3.2 Поля

7.3.2.1 `ADC_Average_TypeDef ADC_Init_TypeDef::ADC_Average`

Количество измерений, используемых для получения результата преобразования. Параметр может принимать любое значение из [ADC_Average_TypeDef](#).

См. определение в файле `niietcm4_adc.h` строка 586

Используется в `ADC_Init()` и `ADC_StructInit()`.

7.3.2.2 `ADC_Measure_TypeDef ADC_Init_TypeDef::ADC_Measure_A`

Определяет режим измерения по каналу A: однополярный и дифференциальный (A-B). Параметр может принимать любое значение из [ADC_Measure_TypeDef](#).

См. определение в файле `niietcm4_adc.h` строка 580

Используется в `ADC_Init()` и `ADC_StructInit()`.

7.3.2.3 `ADC_Measure_TypeDef ADC_Init_TypeDef::ADC_Measure_B`

Определяет режим измерения по каналу B: однополярный и дифференциальный (B-A). Параметр может принимать любое значение из [ADC_Measure_TypeDef](#).

См. определение в файле `niietcm4_adc.h` строка 582

Используется в `ADC_Init()` и `ADC_StructInit()`.

7.3.2.4 ADC_Mode_TypeDef ADC_Init_TypeDef::ADC_Mode

Определяет режим работы модуля АЦП. Параметр может принимать любое значение из [ADC_Mode_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 588

Используется в ADC_Init() и ADC_StructInit().

7.3.2.5 uint32_t ADC_Init_TypeDef::ADC_Phase

Фазовая задержка начала преобразования модулем АЦП после запуска модуля секвенсором. Параметр может принимать любое значение из диапазона 0 - 4095.

См. определение в файле niietcm4_adc.h строка 584

Используется в ADC_Init() и ADC_StructInit().

7.3.2.6 ADC_Resolution_TypeDef ADC_Init_TypeDef::ADC_Resolution

Определяет разрядность модуля АЦП. Параметр может принимать любое значение из [ADC_Resolution_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 578

Используется в ADC_Init() и ADC_StructInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_adc.h](#)

7.4 Структура ADC_SEQ_Init_TypeDef

Структура инициализации секвенсоров.

```
#include <niietcm4_adc.h>
```

Поля данных

- [ADC_SEQ_StartEvent_TypeDef ADC_SEQ_StartEvent](#)
- [FunctionalState ADC_SEQ_SWReqEn](#)
- [uint32_t ADC_Channels](#)
- [uint32_t ADC_SEQ_ConversionCount](#)
- [uint32_t ADC_SEQ_ConversionDelay](#)
- [uint32_t ADC_SEQ_DC](#)

7.4.1 Подробное описание

Структура инициализации секвенсоров.

См. определение в файле niietcm4_adc.h строка 617

7.4.2 Поля

7.4.2.1 uint32_t ADC_SEQ_Init_TypeDef::ADC_Channels

Выбор каналов для измерений. Параметр может принимать любую совокупность значений из [Маски каналов для измерений](#).

См. определение в файле niietcm4_adc.h строка 623

Используется в ADC_SEQ_Init() и ADC_SEQ_StructInit().

7.4.2.2 uint32_t ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount

Задание количества перезапусков модулей АЦП секвенсором после его запуска по событию. Параметр может принимать любое значение из диапазона 1 - 256.

См. определение в файле niietcm4_adc.h строка 625

Используется в ADC_SEQ_Init() и ADC_SEQ_StructInit().

7.4.2.3 uint32_t ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay

Задание задержки запуска модуля АЦП. Параметр может принимать любое значение из диапазона 0x00000000 - 0x00FFFFFF.

См. определение в файле niietcm4_adc.h строка 627

Используется в ADC_SEQ_Init() и ADC_SEQ_StructInit().

7.4.2.4 uint32_t ADC_SEQ_Init_TypeDef::ADC_SEQ_DC

Разрешение работы цифрового компаратора секвенсором. Параметр может принимать любую совокупность значений из [Маски выбора цифровых компараторов](#).

См. определение в файле niietcm4_adc.h строка 629

Используется в ADC_SEQ_Init() и ADC_SEQ_StructInit().

7.4.2.5 ADC_SEQ_StartEvent_TypeDef ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent

Определяет событие запуска секвенсора. Параметр может принимать любое значение из [ADC_SEQ_StartEvent_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 619

Используется в ADC_SEQ_Init() и ADC_SEQ_StructInit().

7.4.2.6 FunctionalState ADC_SEQ_Init_TypeDef::ADC_SEQ_SWReqEn

Разрешает секвенсору запускаться по программному запросу. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_adc.h строка 621

Используется в ADC_SEQ_Init() и ADC_SEQ_StructInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_adc.h](#)

7.5 Структура CAP_Capture_Init_TypeDef

Структура инициализации режима захвата.

```
#include <niietcm4_cap.h>
```

Поля данных

- `uint32_t CAP_Capture_Prescale`
- `CAP_Capture_Mode_TypeDef CAP_CaptureMode`
- `uint32_t CAP_Capture_StopVal`
- `FunctionalState CAP_Capture_RstEvent0`
- `FunctionalState CAP_Capture_RstEvent1`
- `FunctionalState CAP_Capture_RstEvent2`
- `FunctionalState CAP_Capture_RstEvent3`
- `CAP_Capture_Polarity_TypeDef CAP_Capture_PolarityEvent0`
- `CAP_Capture_Polarity_TypeDef CAP_Capture_PolarityEvent1`
- `CAP_Capture_Polarity_TypeDef CAP_Capture_PolarityEvent2`
- `CAP_Capture_Polarity_TypeDef CAP_Capture_PolarityEvent3`

7.5.1 Подробное описание

Структура инициализации режима захвата.

См. определение в файле `niietcm4_cap.h` строка 178

7.5.2 Поля

7.5.2.1 `CAP_Capture_Polarity_TypeDef CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`

Определяет фронт события захвата 0. Параметр может принимать любое значение из `CAP_Polarity_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 194

Используется в `CAP_Capture_Init()` и `CAP_Capture_StructInit()`.

7.5.2.2 `CAP_Capture_Polarity_TypeDef CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent1`

Определяет фронт события захвата 1. Параметр может принимать любое значение из `CAP_Polarity_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 196

Используется в `CAP_Capture_Init()` и `CAP_Capture_StructInit()`.

7.5.2.3 `CAP_Capture_Polarity_TypeDef CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent2`

Определяет фронт события захвата 2. Параметр может принимать любое значение из `CAP_Polarity_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 198

Используется в `CAP_Capture_Init()` и `CAP_Capture_StructInit()`.

7.5.2.4 `CAP_Capture_Polarity_TypeDef CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`

Определяет фронт события захвата 3. Параметр может принимать любое значение из `CAP_Polarity_TypeDef`.

См. определение в файле `niietcm4_cap.h` строка 200

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.5 uint32_t CAP_Capture_Init_TypeDef::CAP_Capture_Prescale

Предварительный делитель событий. Параметр может принимать любое значение из диапазона 0-63. 0 - делитель выключен.

См. определение в файле niietcm4_cap.h строка 180

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.6 FunctionalState CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent0

Определяет сброс таймера после события захвата 0. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_cap.h строка 186

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.7 FunctionalState CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1

Определяет сброс таймера после события захвата 1. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_cap.h строка 188

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.8 FunctionalState CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent2

Определяет сброс таймера после события захвата 2. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_cap.h строка 190

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.9 FunctionalState CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3

Определяет сброс таймера после события захвата 3. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_cap.h строка 192

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.10 uint32_t CAP_Capture_Init_TypeDef::CAP_Capture_StopVal

Значение счетчика событий для остановки одиночного режима захвата. Параметр может принимать любое значение из диапазона 0-3.

См. определение в файле niietcm4_cap.h строка 184

Используется в CAP_Capture_Init() и CAP_Capture_StructInit().

7.5.2.11 CAP_Capture_Mode_TypeDef CAP_Capture_Init_TypeDef::CAP_CaptureMode

Определяет режим работы захвата. Параметр может принимать любое значение из [CAP_Capture_Mode_TypeDef](#).

См. определение в файле `niietcm4_cap.h` строка 182

Используется в `CAP_Capture_Init()` и `CAP_Capture_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_cap.h](#)

7.6 Структура `CAP_Init_TypeDef`

Структура инициализации блока захвата в целом.

`#include <niietcm4_cap.h>`

Поля данных

- [CAP_Halt_TypeDef CAP_Halt](#)
- [FunctionalState CAP_SyncCmd](#)
- [CAP_SyncOut_TypeDef CAP_SyncOut](#)
- [CAP_Mode_TypeDef CAP_Mode](#)

7.6.1 Подробное описание

Структура инициализации блока захвата в целом.

См. определение в файле `niietcm4_cap.h` строка 162

7.6.2 Поля

7.6.2.1 `CAP_Halt_TypeDef CAP_Init_TypeDef::CAP_Halt`

Выбор режима остановки таймера при отладке. Параметр может принимать любое значение из [CAP_Halt_TypeDef](#).

См. определение в файле `niietcm4_cap.h` строка 164

Используется в `CAP_Init()` и `CAP_StructInit()`.

7.6.2.2 `CAP_Mode_TypeDef CAP_Init_TypeDef::CAP_Mode`

Выбор режима работы блока захвата. Параметр может принимать любое значение из [CAP_Mode_TypeDef](#).

См. определение в файле `niietcm4_cap.h` строка 170

Используется в `CAP_Init()` и `CAP_StructInit()`.

7.6.2.3 `FunctionalState CAP_Init_TypeDef::CAP_SyncCmd`

Определяет возможность синхронизации. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_cap.h` строка 166

Используется в `CAP_Init()` и `CAP_StructInit()`.

7.6.2.4 CAP_SyncOut_TypeDef CAP_Init_TypeDef::CAP_SyncOut

Выбор источника выходного сигнала синхронизации. Параметр может принимать любое значение из [CAP_SyncOut_TypeDef](#).

См. определение в файле `niietcm4_cap.h` строка 168

Используется в `CAP_Init()` и `CAP_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_cap.h](#)

7.7 Структура CAP_PWM_Init_TypeDef

Структура инициализации режима ШИМ.

```
#include <niietcm4_cap.h>
```

Поля данных

- `uint32_t CAP_PWM_Period`
- `uint32_t CAP_PWM_Compare`
- `CAP_PWM_Polarity_TypeDef CAP_PWM_Polarity`

7.7.1 Подробное описание

Структура инициализации режима ШИМ.

См. определение в файле `niietcm4_cap.h` строка 221

7.7.2 Поля

7.7.2.1 `uint32_t CAP_PWM_Init_TypeDef::CAP_PWM_Compare`

Значение сравнения ШИМ. Параметр может принимать любое значение из диапазона `0x00000000-0xFFFFFFFF`.

См. определение в файле `niietcm4_cap.h` строка 225

Используется в `CAP_PWM_Init()` и `CAP_PWM_StructInit()`.

7.7.2.2 `uint32_t CAP_PWM_Init_TypeDef::CAP_PWM_Period`

Значение периода ШИМ. Параметр может принимать любое значение из диапазона `0x00000000-0xFFFFFFFF`.

См. определение в файле `niietcm4_cap.h` строка 223

Используется в `CAP_PWM_Init()` и `CAP_PWM_StructInit()`.

7.7.2.3 `CAP_PWM_Polarity_TypeDef CAP_PWM_Init_TypeDef::CAP_PWM_Polarity`

Выбор полярности ШИМ сигнала. Параметр может принимать любое значение из [CAP_PWM_Polarity_TypeDef](#).

См. определение в файле `niietcm4_cap.h` строка 227

Используется в `CAP_PWM_Init()` и `CAP_PWM_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_cap.h](#)

7.8 Структура DMA_Channel_TypeDef

Тип, описывающий структуру канала DMA.

```
#include <niietcm4_dma.h>
```

Поля данных

- [uint32_t SRC_DATA_END](#)
- [uint32_t DST_DATA_END](#)
- [union {
 \[uint32_t CHANNEL_CFG\]\(#\)
 \[_CHANNEL_CFG_bits CHANNEL_CFG_bit\]\(#\)
};](#)
- [uint32_t RESERVED](#)

7.8.1 Подробное описание

Тип, описывающий структуру канала DMA.

См. определение в файле [niietcm4_dma.h](#) строка 228

7.8.2 Поля

7.8.2.1 [uint32_t DMA_Channel_TypeDef::CHANNEL_CFG](#)

Настройка канала

См. определение в файле [niietcm4_dma.h](#) строка 234

Используется в [DMA_ChannelDeInit\(\)](#).

7.8.2.2 [_CHANNEL_CFG_bits DMA_Channel_TypeDef::CHANNEL_CFG_bit](#)

Настройка канала: побитовый доступ

См. определение в файле [niietcm4_dma.h](#) строка 235

Используется в [DMA_ChannelInit\(\)](#).

7.8.2.3 [uint32_t DMA_Channel_TypeDef::DST_DATA_END](#)

Адрес конца данных приемника

См. определение в файле [niietcm4_dma.h](#) строка 231

Используется в [DMA_ChannelDeInit\(\)](#) и [DMA_ChannelInit\(\)](#).

7.8.2.4 [uint32_t DMA_Channel_TypeDef::SRC_DATA_END](#)

Адрес конца данных источника

См. определение в файле niietcm4_dma.h строка 230

Используется в DMA_ChannelDeInit() и DMA_ChannelInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.9 Структура DMA_ChannelInit_TypeDef

Структура инициализации канала DMA.

```
#include <niietcm4_dma.h>
```

Поля данных

- `uint32_t * DMA_SrcDataEndPtr`
- `uint32_t * DMA_DstDataEndPtr`
- `DMA_Mode_TypeDef DMA_Mode`
- `FunctionalState DMA_NextUseburst`
- `uint32_t DMA_TransfersTotal`
- `DMA_ArbitrationRate_TypeDef DMA_ArbitrationRate`
- `DMA_Protect_TypeDef DMA_SrcProtect`
- `DMA_Protect_TypeDef DMA_DstProtect`
- `DMA_DataSize_TypeDef DMA_SrcDataSize`
- `DMA_DataSize_TypeDef DMA_DstDataSize`
- `DMA_DataInc_TypeDef DMA_SrcDataInc`
- `DMA_DataInc_TypeDef DMA_DstDataInc`

7.9.1 Подробное описание

Структура инициализации канала DMA.

См. определение в файле niietcm4_dma.h строка 383

7.9.2 Поля

7.9.2.1 DMA_ArbitrationRate_TypeDef DMA_ChannelInit_TypeDef::DMA_ArbitrationRate

Выбор количества передач до выполнения переарбитрации. Параметр может принимать любое значение из [DMA_ArbitrationRate_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 393

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.2 uint32_t* DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr

Указатель конца данных приемника.

См. определение в файле niietcm4_dma.h строка 386

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.3 DMA_DataInc_TypeDef DMA_ChannelInit_TypeDef::DMA_DstDataInc

Шаг инкремента адреса приемника при записи Параметр может принимать любое значение из [DMA_DataInc_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 405

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.4 DMA_DataSize_TypeDef DMA_ChannelInit_TypeDef::DMA_DstDataSize

Разрядность данных приемника Параметр может принимать любое значение из [DMA_DataSize_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 401

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.5 DMA_Protect_TypeDef DMA_ChannelInit_TypeDef::DMA_DstProtect

Защита шины при записи в приемник через DMA Параметр может принимать любое значение из [DMA_Protect_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 397

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.6 DMA_Mode_TypeDef DMA_ChannelInit_TypeDef::DMA_Mode

Выбор режима работы DMA. Параметр может принимать любое значение из [DMA_Mode_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 387

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.7 FunctionalState DMA_ChannelInit_TypeDef::DMA_NextUseburst

Контроль установки соответствующего каналу бита в регистре NT_DMA->CHNL_USEBURST_ ← SET. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_dma.h строка 389

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.8 uint32_t* DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr

Указатель конца данных источника.

См. определение в файле niietcm4_dma.h строка 385

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.9 DMA_DataInc_TypeDef DMA_ChannelInit_TypeDef::DMA_SrcDataInc

Шаг инкремента адреса источника при чтении Параметр может принимать любое значение из [DMA_DataInc_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 403

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.10 DMA_DataSize_TypeDef DMA_ChannelInit_TypeDef::DMA_SrcDataSize

Разрядность данных источника Параметр может принимать любое значение из [DMA_DataSize_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 399

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.11 DMA_Protect_TypeDef DMA_ChannelInit_TypeDef::DMA_SrcProtect

Защита шины при чтении из источника через DMA Параметр может принимать любое значение из [DMA_Protect_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 395

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

7.9.2.12 uint32_t DMA_ChannelInit_TypeDef::DMA_TransfersTotal

Общее количество передач DMA. Параметр может принимать любое значение из диапазона 1-1024

См. определение в файле niietcm4_dma.h строка 391

Используется в DMA_ChannelInit() и DMA_ChannelStructInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.10 Структура DMA_ConfigData_TypeDef

Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ.

```
#include <niietcm4_dma.h>
```

Поля данных

- [DMA_ConfigStruct_TypeDef PRM_DATA](#)
- uint32_t [RESERVED0](#) [32]
- [DMA_ConfigStruct_TypeDef ALT_DATA](#)
- uint32_t [RESERVED1](#) [32]

7.10.1 Подробное описание

Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ.

Внимание

Экземпляры этой структуры требуют обязательного выравнивания по 1024 байтам в адресном пространстве! Разрешенные адреса: 0xFFFFX000, 0xFFFFX400, 0xFFFFX800, 0xFFFFX↵C00.

См. определение в файле niietcm4_dma.h строка 256

7.10.2 Поля

7.10.2.1 DMA_ConfigStruct_TypeDef DMA_ConfigData_TypeDef::ALT_DATA

Альтернативная управляющая структура

См. определение в файле niietcm4_dma.h строка 260

7.10.2.2 DMA_ConfigStruct_TypeDef DMA_ConfigData_TypeDef::PRM_DATA

Основная управляющая структура

См. определение в файле niietcm4_dma.h строка 258

7.10.2.3 uint32_t DMA_ConfigData_TypeDef::RESERVED0[32]

Пустышка для неиспользованных 8 каналов DMA

См. определение в файле niietcm4_dma.h строка 259

7.10.2.4 uint32_t DMA_ConfigData_TypeDef::RESERVED1[32]

Пустышка для неиспользованных 8 каналов DMA

См. определение в файле niietcm4_dma.h строка 261

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.11 Структура DMA_ConfigStruct_TypeDef

Управляющая структура данных DMA.

```
#include <niietcm4_dma.h>
```

Поля данных

- [DMA_Channel_TypeDef CH](#) [24]

7.11.1 Подробное описание

Управляющая структура данных DMA.

См. определение в файле niietcm4_dma.h строка 244

7.11.2 Поля

7.11.2.1 DMA_Channel_TypeDef DMA_ConfigStruct_TypeDef::CH[24]

Совокупность из общего количества каналов DMA

См. определение в файле niietcm4_dma.h строка 246

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.12 Структура DMA_Init_TypeDef

Структура инициализации контроллера DMA.

```
#include <niietcm4_dma.h>
```

Поля данных

- [uint32_t DMA_Channel](#)
- [DMA_Protect_TypeDef DMA_Protection](#)
- [FunctionalState DMA_UseBurst](#)
- [FunctionalState DMA_ReqMask](#)
- [FunctionalState DMA_PrmAlt](#)
- [FunctionalState DMA_HighPriority](#)
- [FunctionalState DMA_ChannelEnable](#)

7.12.1 Подробное описание

Структура инициализации контроллера DMA.

См. определение в файле niietcm4_dma.h строка 419

7.12.2 Поля

7.12.2.1 uint32_t DMA_Init_TypeDef::DMA_Channel

Определяет каналы, которые будут настроены. Параметр может принимать любое значение любой или комбинации масок из [Маски каналов по номеру](#).

См. определение в файле niietcm4_dma.h строка 421

Используется в DMA_Init() и DMA_StructInit().

7.12.2.2 FunctionalState DMA_Init_TypeDef::DMA_ChannelEnable

Разрешение работы каналов DMA. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_dma.h строка 432

Используется в DMA_Init() и DMA_StructInit().

7.12.2.3 FunctionalState DMA_Init_TypeDef::DMA_HighPriority

Установка высокого приоритета каналов DMA. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_dma.h строка 430

Используется в DMA_Init() и DMA_StructInit().

7.12.2.4 FunctionalState DMA_Init_TypeDef::DMA_PrmAlt

Установка первичной/альтернативной управляющей структуры каналов DMA. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_dma.h строка 428

Используется в DMA_Init() и DMA_StructInit().

7.12.2.5 DMA_Protect_TypeDef DMA_Init_TypeDef::DMA_Protection

Управление защитой шины при обращении DMA к управляющим данным.

См. определение в файле `niietcm4_dma.h` строка 423

Используется в `DMA_Init()` и `DMA_StructInit()`.

7.12.2.6 FunctionalState DMA_Init_TypeDef::DMA_ReqMask

Маскирование (игнорирование) запросов от периферии на обслуживание каналов DMA. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_dma.h` строка 426

Используется в `DMA_Init()` и `DMA_StructInit()`.

7.12.2.7 FunctionalState DMA_Init_TypeDef::DMA_UseBurst

Установка пакетного обмена каналов DMA. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_dma.h` строка 424

Используется в `DMA_Init()` и `DMA_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.13 Структура DMA_Protect_TypeDef

Защита шины при чтении из источника или записи в приемник через DMA.

```
#include <niietcm4_dma.h>
```

Поля данных

- [FunctionalState PRIVELGED](#)
- [FunctionalState BUFFERABLE](#)
- [FunctionalState CACHEABLE](#)

7.13.1 Подробное описание

Защита шины при чтении из источника или записи в приемник через DMA.

См. определение в файле `niietcm4_dma.h` строка 332

7.13.2 Поля

7.13.2.1 FunctionalState DMA_Protect_TypeDef::BUFFERABLE

Управление буферизацией доступа

См. определение в файле `niietcm4_dma.h` строка 335

Используется в `DMA_ChannelInit()`, `DMA_ChannelStructInit()`, `DMA_ProtectionConfig()` и `DMA_↔_StructInit()`.

7.13.2.2 FunctionalState DMA_Protect_TypeDef::CACHEABLE

Управление кэшированием доступа

См. определение в файле `niietcm4_dma.h` строка 336

Используется в `DMA_ChannelInit()`, `DMA_ChannelStructInit()`, `DMA_ProtectionConfig()` и `DMA_C↔_StructInit()`.

7.13.2.3 FunctionalState DMA_Protect_TypeDef::PRIVELGED

Управление привелегированным доступом

См. определение в файле `niietcm4_dma.h` строка 334

Используется в `DMA_ChannelInit()`, `DMA_ChannelStructInit()`, `DMA_ProtectionConfig()` и `DMA_C↔_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.14 Структура EXTMEM_Init_TypeDef

Структура инициализации внешней памяти.

```
#include <niietcm4_extmem.h>
```

Поля данных

- [EXTMEM_Width_TypeDef EXTMEM_Width](#)
- [EXTMEM_RWWaitState_TypeDef EXTMEM_RWWaitState](#)
- [EXTMEM_WriteWaitState_TypeDef EXTMEM_WriteWaitState](#)
- [EXTMEM_ReadWaitState_TypeDef EXTMEM_ReadWaitState](#)
- `uint32_t CEMask`

7.14.1 Подробное описание

Структура инициализации внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 193

7.14.2 Поля

7.14.2.1 uint32_t EXTMEM_Init_TypeDef::CEMask

Маска адреса для генерации сигналов `Cen` и `Oen`. Параметр может принимать любое значение из диапазона 0-511.

См. определение в файле `niietcm4_extmem.h` строка 203

Используется в `EXTMEM_StructInit()`.

7.14.2.2 EXTMEM_ReadWaitState_TypeDef EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState

Длительность цикла чтения слова данных в системных тактах. Параметр может принимать любое значение из [EXTMEM_ReadWaitState_TypeDef](#).

См. определение в файле `niietcm4_extmem.h` строка 201

Используется в `EXTMEM_StructInit()`.

7.14.2.3 `EXTMEM_RWWaitState_TypeDef` `EXTMEM_Init_TypeDef::EXTMEM_RWWaitState`

Длительность цикла переключения шины в системных тактах. Параметр может принимать любое значение из `EXTMEM_RWWaitState_TypeDef`.

См. определение в файле `niietcm4_extmem.h` строка 197

Используется в `EXTMEM_StructInit()`.

7.14.2.4 `EXTMEM_Width_TypeDef` `EXTMEM_Init_TypeDef::EXTMEM_Width`

Разрядность контроллера внешней памяти. Параметр может принимать любое значение из `EXTMEM_Width_TypeDef`.

См. определение в файле `niietcm4_extmem.h` строка 195

Используется в `EXTMEM_StructInit()`.

7.14.2.5 `EXTMEM_WriteWaitState_TypeDef` `EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState`

Длительность цикла записи слова данных в системных тактах. Параметр может принимать любое значение из `EXTMEM_WriteWaitState_TypeDef`.

См. определение в файле `niietcm4_extmem.h` строка 199

Используется в `EXTMEM_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- `niietcm4_extmem.h`

7.15 Структура `GPIO_Init_TypeDef`

Структура инициализации GPIO.

```
#include <niietcm4_gpio.h>
```

Поля данных

- `uint32_t GPIO_Pin`
- `GPIO_Dir_TypeDef GPIO_Dir`
- `GPIO_Out_TypeDef GPIO_Out`
- `GPIO_Mode_TypeDef GPIO_Mode`
- `GPIO_AltFunc_TypeDef GPIO_AltFunc`
- `GPIO_Load_TypeDef GPIO_Load`
- `GPIO_OutMode_TypeDef GPIO_OutMode`
- `GPIO_PullUp_TypeDef GPIO_PullUp`

7.15.1 Подробное описание

Структура инициализации GPIO.

См. определение в файле `niietcm4_gpio.h` строка 284

7.15.2 Поля

7.15.2.1 GPIO_AltFunc_TypeDef GPIO_Init_TypeDef::GPIO_AltFunc

Определяет номер альтернативной функции выбранных пинов. Параметр может принимать любое значение из [GPIO_AltFunc_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 294

Используется в `GPIO_Init()`, `GPIO_StructInit()` и `RCC_SysClkDiv2Out()`.

7.15.2.2 GPIO_Dir_TypeDef GPIO_Init_TypeDef::GPIO_Dir

Определяет направление работы выбранных пинов. Параметр может принимать любое значение из [GPIO_Dir_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 288

Используется в `GPIO_Init()`, `GPIO_StructInit()` и `RCC_SysClkDiv2Out()`.

7.15.2.3 GPIO_Load_TypeDef GPIO_Init_TypeDef::GPIO_Load

Определяет максисальную нагрузку выбранных пинов. Параметр может принимать любое значение из [GPIO_Load_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 296

Используется в `GPIO_Init()` и `GPIO_StructInit()`.

7.15.2.4 GPIO_Mode_TypeDef GPIO_Init_TypeDef::GPIO_Mode

Определяет режим работы выбранных пинов. Параметр может принимать любое значение из [GPIO_Mode_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 292

Используется в `GPIO_Init()`, `GPIO_StructInit()` и `RCC_SysClkDiv2Out()`.

7.15.2.5 GPIO_Out_TypeDef GPIO_Init_TypeDef::GPIO_Out

Определяет включение выхода выбранных пинов. Параметр может принимать любое значение из [GPIO_Out_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 290

Используется в `GPIO_Init()`, `GPIO_StructInit()` и `RCC_SysClkDiv2Out()`.

7.15.2.6 GPIO_OutMode_TypeDef GPIO_Init_TypeDef::GPIO_OutMode

Определяет режим работы выходных каскадов выбранных пинов. Параметр может принимать любое значение из [GPIO_OutMode_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 298

Используется в `GPIO_Init()` и `GPIO_StructInit()`.

7.15.2.7 uint32_t GPIO_Init_TypeDef::GPIO_Pin

Определяет пины, которые будут настроены. Параметр может принимать любое значение из [Маски пинов](#).

См. определение в файле `niietcm4_gpio.h` строка 286

Используется в `GPIO_Init()`, `GPIO_StructInit()` и `RCC_SysClkDiv2Out()`.

7.15.2.8 `GPIO_PullUp_TypeDef` `GPIO_Init_TypeDef::GPIO_PullUp`

Определяет включение внутренней подтяжки к питанию выбранных пинов. Параметр может принимать любое значение из [GPIO_PullUp_TypeDef](#).

См. определение в файле `niietcm4_gpio.h` строка 300

Используется в `GPIO_Init()` и `GPIO_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_gpio.h](#)

7.16 Структура `RCC_PLLInit_TypeDef`

Структура инициализации PLL.

```
#include <niietcm4_rcc.h>
```

Поля данных

- `uint32_t` [RCC_PLLDiv](#)
- [RCC_PLLRef_TypeDef](#) [RCC_PLLRef](#)
- [RCC_PLLNO_TypeDef](#) [RCC_PLLNO](#)
- `uint32_t` [RCC_PLLNR](#)
- `uint32_t` [RCC_PLLNF](#)

7.16.1 Подробное описание

Структура инициализации PLL.

См. определение в файле `niietcm4_rcc.h` строка 376

7.16.2 Поля

7.16.2.1 `uint32_t` `RCC_PLLInit_TypeDef::RCC_PLLDiv`

Значение делителя сигнала на выходе блока PLL. Параметр может принимать любое значение из диапазона 0-255.

См. определение в файле `niietcm4_rcc.h` строка 378

Используется в `RCC_PLLAutoConfig()`, `RCC_PLLInit()` и `RCC_PLLStructInit()`.

7.16.2.2 `uint32_t` `RCC_PLLInit_TypeDef::RCC_PLLNF`

Делитель обратной связи NF. Параметр может принимать любое значение из иапазона 2-513.

См. определение в файле `niietcm4_rcc.h` строка 386

Используется в `RCC_PLLAutoConfig()`, `RCC_PLLInit()` и `RCC_PLLStructInit()`.

7.16.2.3 RCC_PLLNO_TypeDef RCC_PLLInit_TypeDef::RCC_PLLNO

Выходной делитель NO. Параметр может принимать любое значение из [RCC_PLLNO_TypeDef](#).

См. определение в файле niietcm4_rcc.h строка 382

Используется в `RCC_PLLAutoConfig()`, `RCC_PLLInit()` и `RCC_PLLStructInit()`.

7.16.2.4 uint32_t RCC_PLLInit_TypeDef::RCC_PLLNR

Опорный делитель NR. Параметр может принимать любое значение из иапазона 2-33.

См. определение в файле niietcm4_rcc.h строка 384

Используется в `RCC_PLLAutoConfig()`, `RCC_PLLInit()` и `RCC_PLLStructInit()`.

7.16.2.5 RCC_PLLRef_TypeDef RCC_PLLInit_TypeDef::RCC_PLLRef

Источник опорного сигнала PLL. Параметр может принимать любое значение из [RCC_PLLRef_TypeDef](#).

См. определение в файле niietcm4_rcc.h строка 380

Используется в `RCC_PLLAutoConfig()`, `RCC_PLLInit()` и `RCC_PLLStructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_rcc.h](#)

7.17 Структура RTC_Date_TypeDef

Структура даты.

```
#include <niietcm4_rtc.h>
```

Поля данных

- [RTC_Weekday_TypeDef](#) [RTC_Weekday](#)
- [uint32_t](#) [RTC_Day](#)
- [uint32_t](#) [RTC_Month](#)
- [uint32_t](#) [RTC_Year](#)

7.17.1 Подробное описание

Структура даты.

См. определение в файле niietcm4_rtc.h строка 206

7.17.2 Поля

7.17.2.1 uint32_t RTC_Date_TypeDef::RTC_Day

Значение дней. Если выбран бинарный формат [RTC_Format_BIN](#), то допустимые значения от 1 до 31. Если выбран двоично-десятичный формат [RTC_Format_BCD](#), то допустимые значения 0x01-0x09, 0x10-0x19, 0x20-0x29, 0x30-0x31.

См. определение в файле niietcm4_rtc.h строка 210

7.17.2.2 uint32_t RTC_Date_TypeDef::RTC_Month

Значение месяцев. Если выбран бинарный формат [RTC_Format_BIN](#), то допустимые значения от 1 до 12. Если выбран двоично-десятичный формат [RTC_Format_BCD](#), то допустимые значения 0x01-0x09, 0x10-0x12.

См. определение в файле `niietcm4_rtc.h` строка 215

7.17.2.3 RTC_Weekday_TypeDef RTC_Date_TypeDef::RTC_Weekday

Значение дней недели. Параметр может принимать любое значение из [RTC_Weekday_TypeDef](#).

См. определение в файле `niietcm4_rtc.h` строка 208

7.17.2.4 uint32_t RTC_Date_TypeDef::RTC_Year

Значение лет. Если выбран бинарный формат [RTC_Format_BIN](#), то допустимые значения от 0 до 99. Если выбран двоично-десятичный формат [RTC_Format_BCD](#), то допустимые значения 0x00-0x09, 0x10-0x19, 0x20-0x29, 0x30-0x39, 0x40-0x49, 0x50-0x59, 0x60-0x69, 0x70-0x79, 0x80-0x89, 0x90-0x99.

См. определение в файле `niietcm4_rtc.h` строка 220

Объявления и описания членов структуры находятся в файле:

- [niietcm4_rtc.h](#)

7.18 Структура RTC_Time_TypeDef

Структура времени.

```
#include <niietcm4_rtc.h>
```

Поля данных

- uint32_t [RTC_Psecond](#)
- uint32_t [RTC_Second](#)
- uint32_t [RTC_Minute](#)
- uint32_t [RTC_Hour](#)

7.18.1 Подробное описание

Структура времени.

См. определение в файле `niietcm4_rtc.h` строка 178

7.18.2 Поля

7.18.2.1 uint32_t RTC_Time_TypeDef::RTC_Hour

Значение часов. Если выбран бинарный формат [RTC_Format_BIN](#), то допустимые значения от 0 до 23. Если выбран двоично-десятичный формат [RTC_Format_BCD](#), то допустимые значения 0x00-0x09, 0x10-0x19, 0x20-0x23.

См. определение в файле `niietcm4_rtc.h` строка 194

7.18.2.2 uint32_t RTC_Time_TypeDef::RTC_Minute

Значение минут. Если выбран бинарный формат [RTC_Format_BIN](#), то допустимые значения от 0 до 59. Если выбран двоично-десятичный формат [RTC_Format_BCD](#), то допустимые значения 0x00-0x09, 0x10-0x19, 0x20-0x29, 0x30-0x39, 0x40-0x49, 0x50-0x59.

См. определение в файле `niietcm4_rtc.h` строка 188

7.18.2.3 uint32_t RTC_Time_TypeDef::RTC_Psecond

Значение долей секунд. Параметр может принимать любое значение из диапазона 0-1023.

См. определение в файле `niietcm4_rtc.h` строка 180

7.18.2.4 uint32_t RTC_Time_TypeDef::RTC_Second

Значение секунд. Если выбран бинарный формат [RTC_Format_BIN](#), то допустимые значения от 0 до 59. Если выбран двоично-десятичный формат [RTC_Format_BCD](#), то допустимые значения 0x00-0x09, 0x10-0x19, 0x20-0x29, 0x30-0x39, 0x40-0x49, 0x50-0x59.

См. определение в файле `niietcm4_rtc.h` строка 182

Объявления и описания членов структуры находятся в файле:

- [niietcm4_rtc.h](#)

7.19 Структура UART_Init_TypeDef

Структура инициализации UART.

```
#include <niietcm4_uart.h>
```

Поля данных

- [UART_StopBit_TypeDef UART_StopBit](#)
- [UART_ParityBit_TypeDef UART_ParityBit](#)
- [UART_DataWidth_TypeDef UART_DataWidth](#)
- [uint32_t UART_ClkFreq](#)
- [uint32_t UART_BaudRate](#)
- [UART_FIFOLevel_TypeDef UART_FIFOLevelRx](#)
- [UART_FIFOLevel_TypeDef UART_FIFOLevelTx](#)
- [FunctionalState UART_FIFOEn](#)
- [FunctionalState UART_RxEn](#)
- [FunctionalState UART_TxEn](#)

7.19.1 Подробное описание

Структура инициализации UART.

См. определение в файле `niietcm4_uart.h` строка 285

7.19.2 Поля

7.19.2.1 uint32_t UART_Init_TypeDef::UART_BaudRate

Желаемая скорость передачи данных в бит/с Максимальное значение 921600

См. определение в файле `niietcm4_uart.h` строка 294

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.2 `uint32_t UART_Init_TypeDef::UART_ClkFreq`

Значение текущей частоты в Гц, подведенной к блоку UART

См. определение в файле `niietcm4_uart.h` строка 293

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.3 `UART_DataWidth_TypeDef UART_Init_TypeDef::UART_DataWidth`

Количество передаваемых/принимаемых информационных бит. Параметр может принимать любое значение из [UART_DataWidth_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 291

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.4 `FunctionalState UART_Init_TypeDef::UART_FIFOEn`

Разрешение режима FIFO буфера приемника и передатчика. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_uart.h` строка 300

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.5 `UART_FIFOLevel_TypeDef UART_Init_TypeDef::UART_FIFOLevelRx`

Порог заполнения буфера приемника при срабатывании. Параметр может принимать любое значение из [UART_FIFOLevel_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 296

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.6 `UART_FIFOLevel_TypeDef UART_Init_TypeDef::UART_FIFOLevelTx`

Порог заполнения буфера передатчика при срабатывании. Параметр может принимать любое значение из [UART_FIFOLevel_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 298

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.7 `UART_ParityBit_TypeDef UART_Init_TypeDef::UART_ParityBit`

Выбор режима бита четности. Параметр может принимать любое значение из [UART_ParityBit_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 289

Используется в `UART_Init()` и `UART_StructInit()`.

7.19.2.8 `FunctionalState UART_Init_TypeDef::UART_RxEn`

Разрешение приема. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 302

Используется в UART_Init() и UART_StructInit().

7.19.2.9 UART_StopBit_TypeDef UART_Init_TypeDef::UART_StopBit

Выбор режима передачи стопового бита. Параметр может принимать любое значение из [UART_StopBit_TypeDef](#).

См. определение в файле niietcm4_uart.h строка 287

Используется в UART_Init() и UART_StructInit().

7.19.2.10 FunctionalState UART_Init_TypeDef::UART_TxEn

Разрешение передачи. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 304

Используется в UART_Init() и UART_StructInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_uart.h](#)

7.20 Структура UART_ModemInit_TypeDef

Структура инициализации модемного режима.

```
#include <niietcm4_uart.h>
```

Поля данных

- [FunctionalState UART_InvDTR](#)
- [FunctionalState UART_InvRTS](#)
- [FunctionalState UART_RTSEn](#)
- [FunctionalState UART_CTSEn](#)

7.20.1 Подробное описание

Структура инициализации модемного режима.

См. определение в файле niietcm4_uart.h строка 269

7.20.2 Поля

7.20.2.1 FunctionalState UART_ModemInit_TypeDef::UART_CTSEn

Разрешение аппаратного управления потоком данных по линии CTS. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 277

Используется в UART_ModemConfig() и UART_ModemStructInit().

7.20.2.2 FunctionalState UART_ModemInit_TypeDef::UART_InvDTR

Управление инверсией сигнала на линии состояния модема DTR. Выключенная инверсия означает высокий уровень сигнала. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_uart.h` строка 271

Используется в `UART_ModemConfig()` и `UART_ModemStructInit()`.

7.20.2.3 FunctionalState UART_ModemInit_TypeDef::UART_InvRTS

Управление инверсией сигнала на линии состояния модема RTS. Выключенная инверсия означает высокий уровень сигнала. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_uart.h` строка 273

Используется в `UART_ModemConfig()` и `UART_ModemStructInit()`.

7.20.2.4 FunctionalState UART_ModemInit_TypeDef::UART_RTSEn

Разрешение аппаратного управления потоком данных по линии RTS. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_uart.h` строка 275

Используется в `UART_ModemConfig()` и `UART_ModemStructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_uart.h](#)

Глава 8

Файлы

8.1 Файл niietcm4.h

Это главный заголовочный файл драйвера, обычно включаемый в main.c.

```
#include "niietcm4_conf.h"
```

Макросы

- `#define EXT_OSC_VALUE ((uint32_t)12000000)`
Определение частоты используемого внешнего тактового генератора.
- `#define INT_OSC_VALUE ((uint32_t)8000000)`
Определение частоты частоты внутреннего тактового генератора.
- `#define SET_BIT(REG, BIT) ((REG) |= (BIT))`
Установить бит в регистре.
- `#define CLEAR_BIT(REG, BIT) ((REG) &= ~(BIT))`
Сбросить бит в регистре.
- `#define READ_BIT(REG, BIT) ((REG) & (BIT))`
Прочитать бит из регистра.
- `#define CLEAR_REG(REG) ((REG) = (0x0))`
Обнулить значение регистра.
- `#define WRITE_REG(REG, VAL) ((REG) = (VAL))`
Записать значение в регистр.
- `#define READ_REG(REG) ((REG))`
Прочитать значение из регистра.
- `#define MODIFY_REG(REG, CLEARMASK, SETMASK) WRITE_REG((REG), (((READ_REG(REG) & (~CLEARMASK))) | (SETMASK)))`
Изменить значение регистра по маске.
- `#define IS_FUNCTIONAL_STATE(STATE) (((STATE) == DISABLE) || ((STATE) == ENABLE))`
Макрос проверки аргументов типа `FunctionalState`.
- `#define IS_TIMER_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_TIMER_TypeDef`.
- `#define IS_GPIO_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_GPIO_TypeDef`.
- `#define IS_UART_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_UART_TypeDef`.

- `#define IS_SPI_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_SPI_TypeDef`.
- `#define IS_CAP_ALL_PERIPH(PERIPH)`
Макрос проверки аргументов типа `NT_CAP_TypeDef`.

Перечисления

- `enum FunctionalState { DISABLE = 0, ENABLE = 1 }`
Описывает логическое состояние периферии. Используется для операций включения/выключения периферийных блоков.
- `enum OperationStatus { OK = 0, ERROR = 1 }`
Описывает коды возврата для функций при выполнении какой-либо операции.
- `enum FlagStatus { Flag_CLEAR = 0, Flag_SET = 1 }`
Описывает возможные состояния флага при запросе его статуса.

8.1.1 Подробное описание

Это главный заголовочный файл драйвера, обычно включаемый в `main.c`.

Этот файл содержит:

- Главный заголовочный файл целевого устройства, с описанием всех регистров его периферии
- Область настройки драйвера, которая позволяет сконфигурировать:
 - Модель целевого устройства
 - Используемые тактовые частоты
- Макросы для доступа к регистрам периферии

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.1.2 Макросы

8.1.2.1 #define EXT_OSC_VALUE ((uint32_t)12000000)

Определение частоты используемого внешнего тактового генератора.

Совет: Чтобы избежать необходимости каждый раз изменять этот файл, можно передать определение устройства компилятору через ключ.

Например, для GCC ARM это выглядит так:

```
-DEXT_OSC_VALUE=16000000
```

Частота внешнего тактового генератора [Гц].

См. определение в файле niietcm4.h строка 73

Используется в RCC_PLLAutoConfig() и UART_StructInit().

8.1.2.2 #define INT_OSC_VALUE ((uint32_t)8000000)

Определение частоты частоты внутреннего тактового генератора.

Конфигурируется автоматически в зависимости от выбранного целевого устройства. Частота внутреннего тактового генератора [Гц].

См. определение в файле niietcm4.h строка 88

8.1.2.3 #define IS_CAP_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_CAP0) || \
((PERIPH) == NT_CAP1) || \
((PERIPH) == NT_CAP2) || \
((PERIPH) == NT_CAP3) || \
((PERIPH) == NT_CAP4) || \
((PERIPH) == NT_CAP5)
```

Макрос проверки аргументов типа NT_CAP_TypeDef.

См. определение в файле niietcm4.h строка 232

Используется в CAP_Capture_Cmd(), CAP_Capture_GetCap0(), CAP_Capture_GetCap1(), CAP_Capture_GetCap2(), CAP_Capture_GetCap3(), CAP_Capture_Init(), CAP_Capture_SetCap0(), CAP_Capture_SetCap1(), CAP_Capture_SetCap2(), CAP_Capture_SetCap3(), CAP_DeInit(), CAP_GetShadowTimer(), CAP_GetTimer(), CAP_Init(), CAP_ITCmd(), CAP_ITForceCmd(), CAP_ITPendClear(), CAP_ITPendStatus(), CAP_ITStatus(), CAP_ITStatusClear(), CAP_PWM_GetCompare(), CAP_PWM_GetPeriod(), CAP_PWM_GetShadowCompare(), CAP_PWM_GetShadowPeriod(), CAP_PWM_Init(), CAP_PWM_SetCompare(), CAP_PWM_SetPeriod(), CAP_PWM_SetShadowCompare(), CAP_PWM_SetShadowPeriod(), CAP_SetShadowTimer(), CAP_SetTimer(), CAP_SwSync(), CAP_SyncCmd() и CAP_TimerCmd().

8.1.2.4 #define IS_GPIO_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_GPIOA) || \
((PERIPH) == NT_GPIOB) || \
```

```
((PERIPH) == NT_GPIOC) || \
((PERIPH) == NT_GPIOD) || \
((PERIPH) == NT_GPIOE) || \
((PERIPH) == NT_GPIOF) || \
((PERIPH) == NT_GPIOG) || \
((PERIPH) == NT_GPIOH))
```

Макрос проверки аргументов типа NT_GPIO_TypeDef.

См. определение в файле niietcm4.h строка 201

Используется в GPIO_ClearBits(), GPIO_DeInit(), GPIO_Init(), GPIO_ITCmd(), GPIO_ITConfig(), GPIO_ITStatusClear(), GPIO_QualCmd(), GPIO_QualConfig(), GPIO_Read(), GPIO_ReadBit(), GPIO_ReadMask(), GPIO_SetBits(), GPIO_SyncCmd(), GPIO_ToggleBits(), GPIO_Write(), GPIO_WriteBit() и GPIO_WriteMask().

8.1.2.5 #define IS_SPI_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_SPI0) || \
((PERIPH) == NT_SPI1) || \
((PERIPH) == NT_SPI2) || \
((PERIPH) == NT_SPI3))
```

Макрос проверки аргументов типа NT_SPI_TypeDef.

См. определение в файле niietcm4.h строка 223

Используется в RCC_SPIClkCmd(), RCC_SPIClkDivConfig() и RCC_SPIClkSel().

8.1.2.6 #define IS_TIMER_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_TIMER0) || \
((PERIPH) == NT_TIMER1) || \
((PERIPH) == NT_TIMER2))
```

Макрос проверки аргументов типа NT_TIMER_TypeDef.

См. определение в файле niietcm4.h строка 193

Используется в TIMER_Cmd(), TIMER_ExtInputConfig(), TIMER_FreqConfig(), TIMER_GetCounter(), TIMER_GetReload(), TIMER_ITCmd(), TIMER_ITStatus(), TIMER_ITStatusClear(), TIMER_PeriodConfig(), TIMER_SetCounter() и TIMER_SetReload().

8.1.2.7 #define IS_UART_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_UART0) || \
((PERIPH) == NT_UART1) || \
((PERIPH) == NT_UART2) || \
((PERIPH) == NT_UART3))
```

Макрос проверки аргументов типа NT_UART_TypeDef.

См. определение в файле niietcm4.h строка 214

Используется в RCC_UARTClkCmd(), RCC_UARTClkDivConfig(), RCC_UARTClkSel(), UART_BaudRateDivConfig(), UART_Break(), UART_Cmd(), UART_DeInit(), UART_DMABlkOnErrCmd(), UART_DMACmd(), UART_ErrorStatus(), UART_ErrorStatusClear(), UART_FlagStatus(), UART_Init(), UART_ITCmd(), UART_ITFIFOLevelConfig(), UART_ITMaskedStatus(), UART_ITRawStatus(), UART_ITStatusClear(), UART_ModemConfig(), UART_RecieveData() и UART_SendData().

8.2 Файл niietcm4_adc.c

Файл содержит реализацию всех функций для работы с модулями АЦП, секвенсорами, цифровыми компараторами.

```
#include "niietcm4_adc.h"
```

Функции

- void `ADC_Cmd` (`ADC_Module_TypeDef` `ADC_Module`, `FunctionalState` `State`)
Включение модуля АЦП.
- void `ADC_DeInit` (`ADC_Module_TypeDef` `ADC_Module`)
Устанавливает все регистры модуля АЦП значениями по умолчанию.
- void `ADC_Init` (`ADC_Module_TypeDef` `ADC_Module`, `ADC_Init_TypeDef` *`ADC_InitStruct`)
Инициализирует выбранный модуль АЦП согласно параметрам структуры `ADC_InitStruct`.
- void `ADC_StructInit` (`ADC_Init_TypeDef` *`ADC_InitStruct`)
Заполнение каждого члена структуры `ADC_InitStruct` значениями по умолчанию.
- void `ADC_DC_DeInit` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`)
Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.
- void `ADC_DC_Init` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `ADC_DC_Init_TypeDef` *`ADC_DC_InitStruct`)
Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры `ADC_DC_InitStruct`.
- void `ADC_DC_StructInit` (`ADC_DC_Init_TypeDef` *`ADC_DC_InitStruct`)
Заполнение каждого члена структуры `ADC_DC_InitStruct` значениями по умолчанию.
- void `ADC_SEQ_DeInit` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`)
Устанавливает все регистры выбранного секвенсора значениями по умолчанию.
- void `ADC_SEQ_Init` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`, `ADC_SEQ_Init_TypeDef` *`ADC_SEQ_InitStruct`)
Инициализирует выбранный секвенсор согласно параметрам структуры `ADC_SEQ_InitStruct`.
- void `ADC_SEQ_StructInit` (`ADC_SEQ_Init_TypeDef` *`ADC_SEQ_InitStruct`)
Заполнение каждого члена структуры `ADC_SEQ_InitStruct` значениями по умолчанию.
- void `ADC_SEQ_DMAConfig` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`, `ADC_SEQ_FIFOLevel_TypeDef` `ADC_SEQ_FIFOLevel`)
Конфигурирует выбранный секвенсор для работы с DMA.
- void `ADC_SEQ_DMACmd` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`, `FunctionalState` `State`)
Включает для выбранного секвенсора генерирование запросов DMA.
- `FlagStatus` `ADC_SEQ_DMAErrorStatus` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`)
Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.
- void `ADC_SEQ_DMAErrorStatusClear` (`ADC_SEQ_Module_TypeDef` `ADC_SEQ_Module`)
Сброс статуса ошибки DMA.
- void `ADC_DC_ITGenCmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)
Разрешает компаратору генерировать сигнал прерывания.
- void `ADC_DC_ITMaskCmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)
Маскирование сигнала прерывания цифрового компаратора.
- void `ADC_DC_ITCmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)

- Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: `ADC_DC_ITGenCmd` и `ADC_DC_ITMaskCmd`.
- `void ADC_DC_ITConfig (ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef ADC_DC_Condition)`
Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.
 - `FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`
Проверка флагов немаскированных прерываний.
 - `FlagStatus ADC_DC_ITMaskedStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`
Проверка флагов маскированных прерываний.
 - `void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)`
Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.
 - `void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)`
Включение прерывания секвенсора.
 - `void ADC_SEQ_ITConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)`
Настройка вызова прерывания секвенсора.
 - `uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.
 - `void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс счетчика прерываний секвенсора.
 - `FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флагов немаскированных прерываний.
 - `FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флагов маскированных прерываний.
 - `void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.
 - `void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)`
Включение секвенсора.
 - `void ADC_SEQ_SWReq ()`
Программный запуск измерений всех разрешенных секвенсоров.
 - `uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Получение результата измерений из буфера секвенсора.
 - `uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.
 - `uint32_t ADC_SEQ_GetFIFOLoad (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Получение количества измерений, сохраненных в буфере секвенсора.
 - `FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.
 - `void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс флага заполнения буфера секвенсора.
 - `FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.
 - `void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс флага пустоты буфера секвенсора.

- void `ADC_DC_Cmd` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`, `FunctionalState` `State`)
Включение выходного триггера цифрового компаратора.
- uint32_t `ADC_DC_GetLastData` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`)
Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.
- `FlagStatus` `ADC_DC_TrigStatus` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`)
Проверка состояния выходного триггера компаратора.
- void `ADC_DC_TrigStatusClear` (`ADC_DC_Module_TypeDef` `ADC_DC_Module`)
Сброс выходного триггера цифрового компаратора.

8.2.1 Подробное описание

Файл содержит реализацию всех функций для работы с модулями АЦП, секвенсорами, цифровыми компараторами.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

10.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.2.2 Функции

8.2.2.1 void `ADC_Cmd` (`ADC_Module_TypeDef` `ADC_Module`, `FunctionalState` `State`)

Включение модуля АЦП.

Аргументы

ADC_Module	Выбор АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 108

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1017

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.3 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 300

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.4 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 1033

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.5 void ADC_DC_Init (ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры ADC_DC_InitStruct.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 319

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Init_TypeDef::ADC_DC_Mode, ADC_DC_Init_TypeDef::ADC_DC_ThresholdHigh, ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow, IS_ADC_DC, IS_ADC_DC_CHANNEL, IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_THRESHOLD.

8.2.2.6 void ADC_DC_ITCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: [ADC_DC_ITGenCmd](#) и [ADC_DC_ITMaskCmd](#).

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 589

Перекрестные ссылки ADC_DC_ITGenCmd(), ADC_DC_ITMaskCmd(), IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.7 void ADC_DC_ITConfig (ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef
ADC_DC_Condition)

Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
ADC_DC_↔ Mode	Режим срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Mode_TypeDef .
ADC_DC_↔ Condition	Условие срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Condition_TypeDef .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 613

Перекрестные ссылки IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_MODULE.

```
8.2.2.8 void ADC_DC_ITGenCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                             FunctionalState State )
```

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 546

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

```
8.2.2.9 void ADC_DC_ITMaskCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                               FunctionalState State )
```

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 563

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

```
8.2.2.10 FlagStatus ADC_DC_ITMaskedStatus ( ADC_DC_Module_TypeDef ADC_DC_Module
                                             )
```

Проверка флагов маскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 655

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.11 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 630

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.12 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 681

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.13 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 342

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Channel_None, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Condition_Low, ADC_DC_Init_↔

_TypeDef::ADC_DC_Mode, ADC_DC_Mode_Single, ADC_DC_Init_TypeDef::ADC_DC_↵ ThresholdHigh и ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow.

8.2.2.14 FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

ADC_DC_↵ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↵ Module_TypeDef .
--------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние триггера.
------------	-----------------------------

См. определение в файле niietcm4_adc.c строка 1051

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.15 void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Сброс выходного триггера цифрового компаратора.

Внимание

Одновременно со сбросом триггеров 0 и 1 компаратора сбрасываются триггеры 10 и 11 компаратора соответственно. То же самое справедливо и для обратного случая. Это происходит аппаратно и программными методами не обходится.

Аргументы

ADC_DC_↵ Module	Выбор цифрового компаратора. Параметр принимает любое значение из AD↵ C_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1079

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

8.2.2.16 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_↵ TypeDef .
------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 123

Перекрестные ссылки ADC_Module_0, ADC_Module_1, ADC_Module_10, ADC_Module_11, A↵ DC_Module_2, ADC_Module_3, ADC_Module_4, ADC_Module_5, ADC_Module_6, ADC_↵ Module_7, ADC_Module_8, ADC_Module_9 и IS_ADC_MODULE.


```
8.2.2.17 void ADC_Init ( ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *  
ADC_InitStruct )
```

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
ADC_InitStruct	Указатель на структуру типа ADC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 199

Перекрестные ссылки [ADC_Init_TypeDef::ADC_Average](#), [ADC_Init_TypeDef::ADC_Measure_A](#), [ADC_Init_TypeDef::ADC_Measure_B](#), [ADC_Init_TypeDef::ADC_Mode](#), [ADC_Module_0](#), [ADC_Module_1](#), [ADC_Module_10](#), [ADC_Module_11](#), [ADC_Module_2](#), [ADC_Module_3](#), [ADC_Module_4](#), [ADC_Module_5](#), [ADC_Module_6](#), [ADC_Module_7](#), [ADC_Module_8](#), [ADC_Module_9](#), [ADC_Init_TypeDef::ADC_Phase](#), [ADC_Init_TypeDef::ADC_Resolution](#), [IS_ADC_AVERAGE](#), [IS_ADC_MEASURE](#), [IS_ADC_MODE](#), [IS_ADC_MODULE](#), [IS_ADC_PHASE](#) и [IS_ADC_RESOLUTION](#).

8.2.2.18 void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение секвенсора.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 848

Перекрестные ссылки [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

8.2.2.19 void ADC_SEQ_DeInit (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

ADC_SEQ_Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 358

Перекрестные ссылки [ADC_SEQ_Module_0](#), [ADC_SEQ_Module_1](#), [ADC_SEQ_Module_2](#), [ADC_SEQ_Module_3](#), [ADC_SEQ_Module_4](#), [ADC_SEQ_Module_5](#), [ADC_SEQ_Module_6](#), [ADC_SEQ_Module_7](#) и [IS_ADC_SEQ_MODULE](#).

8.2.2.20 void ADC_SEQ_DMAMCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 489

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.21 void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из ADC_SEQ_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 472

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

8.2.2.22 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущие состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 505

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.23 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 530

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.24 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 976

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.25 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1001

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.26 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 936

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.27 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 961

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.28 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 898

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.29 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение результата измерений из буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 880

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
8.2.2.30 uint32_t ADC_SEQ_GetFIFOLoad ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)
```

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 916

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.31 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

ITCount	
---------	--

См. определение в файле niietcm4_adc.c строка 750

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.32 void ADC_SEQ_Init (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 418

Перекрестные ссылки ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent, ADC_SEQ_Init_TypeDef::ADC_SEQ_SWReqEn, IS_ADC_CHANNEL, IS_ADC_DC, IS_ADC_SEQ_CONVERSION_COUNT, IS_ADC_SEQ_CONVERSION_DELAY, IS_ADC_SEQ_MODULE, IS_ADC_SEQ_START_EVENT и IS_FUNCTIONAL_STATE.

8.2.2.33 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
FunctionalState State)

Включение прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 697

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.34 void ADC_SEQ_ITConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)

Настройка вызова прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ ITRate	Значение количества перезапусков модулей АЦП секвенсором после которого генерируется прерывание. Параметр принимает любое значение из диапазона 1 - 256.
ADC_SEQ_↔ ITCountSEQRst	Разрешение сброса счетчика прерываний по запуску секвенсора. Если запретить, то счетчик можно будет сбрасывать только программно через ADC_SEQ_IT↔ CountRst . Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 725

Перекрестные ссылки IS_ADC_SEQ_IT_RATE, IS_ADC_SEQ_MODULE и IS_FUNCTIONAL↔
L_STATE.

8.2.2.35 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 768

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.36 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 807

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.37 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 782

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.38 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 832

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.39 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которую необходимо проинициализировать.
-------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 453

Перекрестные ссылки ADC_Channel_None, ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_D↔C_None, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::AD↔C_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef↔::ADC_SEQ_StartEvent, ADC_SEQ_StartEvent_SWReq и ADC_SEQ_Init_TypeDef::ADC_SE↔Q_SWReqEn.

8.2.2.40 void ADC_SEQ_SWReq ()

Программный запуск измерений всех разрешенных секвенсоров.

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 868

8.2.2.41 void ADC_StructInit (ADC_Init_TypeDef * ADC_InitStruct)

Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

Аргументы

ADC_Init↔Struct	Указатель на структуру типа ADC_Init_TypeDef , которую необходимо проинициализировать.
-----------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 283

Перекрестные ссылки ADC_Init_TypeDef::ADC_Average, ADC_Average_Disable, ADC_Init_↔TypeDef::ADC_Measure_A, ADC_Init_TypeDef::ADC_Measure_B, ADC_Measure_Single, ADC↔_Init_TypeDef::ADC_Mode, ADC_Mode_Powerdown, ADC_Init_TypeDef::ADC_Phase, ADC_↔Init_TypeDef::ADC_Resolution и ADC_Resolution_12bit.

8.3 Файл niietcm4_adc.h

Файл содержит все прототипы функций для работы с АЦП, секвенсорами, цифровыми компараторами.

```
#include "niietcm4.h"
```

Структуры данных

- struct [ADC_Init_TypeDef](#)
Структура инициализации модулей АЦП
- struct [ADC_DC_Init_TypeDef](#)
Структура инициализации цифровых компараторов.
- struct [ADC_SEQ_Init_TypeDef](#)
Структура инициализации секвенсоров.

Макросы

```

• #define ADC_Channel_None ((uint32_t)0x00000000)
• #define ADC_Channel_0 ((uint32_t)0x00000001)
• #define ADC_Channel_1 ((uint32_t)0x00000002)
• #define ADC_Channel_2 ((uint32_t)0x00000004)
• #define ADC_Channel_3 ((uint32_t)0x00000008)
• #define ADC_Channel_4 ((uint32_t)0x00000010)
• #define ADC_Channel_5 ((uint32_t)0x00000020)
• #define ADC_Channel_6 ((uint32_t)0x00000040)
• #define ADC_Channel_7 ((uint32_t)0x00000080)
• #define ADC_Channel_8 ((uint32_t)0x00000100)
• #define ADC_Channel_9 ((uint32_t)0x00000200)
• #define ADC_Channel_10 ((uint32_t)0x00000400)
• #define ADC_Channel_11 ((uint32_t)0x00000800)
• #define ADC_Channel_12 ((uint32_t)0x00001000)
• #define ADC_Channel_13 ((uint32_t)0x00002000)
• #define ADC_Channel_14 ((uint32_t)0x00004000)
• #define ADC_Channel_15 ((uint32_t)0x00008000)
• #define ADC_Channel_16 ((uint32_t)0x00010000)
• #define ADC_Channel_17 ((uint32_t)0x00020000)
• #define ADC_Channel_18 ((uint32_t)0x00040000)
• #define ADC_Channel_19 ((uint32_t)0x00080000)
• #define ADC_Channel_20 ((uint32_t)0x00100000)
• #define ADC_Channel_21 ((uint32_t)0x00200000)
• #define ADC_Channel_22 ((uint32_t)0x00400000)
• #define ADC_Channel_23 ((uint32_t)0x00800000)
• #define ADC_Channel_All ((uint32_t)0x0FFFFFFF)
• #define IS_ADC_CHANNEL(CHANNEL) (((CHANNEL) & (uint32_t)0xFF000000) ==
((uint32_t)0x00000000))

```

Макрос проверки попадания масок каналов в допустимый диапазон.

```

• #define ADC_DC_None ((uint32_t)0x00000000)
• #define ADC_DC_0 ((uint32_t)0x00000001)
• #define ADC_DC_1 ((uint32_t)0x00000002)
• #define ADC_DC_2 ((uint32_t)0x00000004)
• #define ADC_DC_3 ((uint32_t)0x00000008)
• #define ADC_DC_4 ((uint32_t)0x00000010)
• #define ADC_DC_5 ((uint32_t)0x00000020)
• #define ADC_DC_6 ((uint32_t)0x00000040)
• #define ADC_DC_7 ((uint32_t)0x00000080)
• #define ADC_DC_8 ((uint32_t)0x00000100)
• #define ADC_DC_9 ((uint32_t)0x00000200)
• #define ADC_DC_10 ((uint32_t)0x00000400)
• #define ADC_DC_11 ((uint32_t)0x00000800)
• #define ADC_DC_12 ((uint32_t)0x00001000)
• #define ADC_DC_13 ((uint32_t)0x00002000)
• #define ADC_DC_14 ((uint32_t)0x00004000)
• #define ADC_DC_15 ((uint32_t)0x00008000)
• #define ADC_DC_16 ((uint32_t)0x00010000)
• #define ADC_DC_17 ((uint32_t)0x00020000)
• #define ADC_DC_18 ((uint32_t)0x00040000)
• #define ADC_DC_19 ((uint32_t)0x00080000)
• #define ADC_DC_20 ((uint32_t)0x00100000)
• #define ADC_DC_21 ((uint32_t)0x00200000)
• #define ADC_DC_22 ((uint32_t)0x00400000)

```

- `#define ADC_DC_23 ((uint32_t)0x00800000)`
- `#define ADC_DC_All ((uint32_t)0x00FFFFFF)`
- `#define IS_ADC_DC(DC) (((DC) & (uint32_t)0xFF000000) == ((uint32_t)0x00000000))`

Макрос проверки попадания масок компараторов в допустимый диапазон.

- `#define ADC_SEQ_0 ((uint32_t)0x00000001)`
- `#define ADC_SEQ_1 ((uint32_t)0x00000002)`
- `#define ADC_SEQ_2 ((uint32_t)0x00000004)`
- `#define ADC_SEQ_3 ((uint32_t)0x00000008)`
- `#define ADC_SEQ_4 ((uint32_t)0x00000010)`
- `#define ADC_SEQ_5 ((uint32_t)0x00000020)`
- `#define ADC_SEQ_6 ((uint32_t)0x00000040)`
- `#define ADC_SEQ_7 ((uint32_t)0x00000080)`
- `#define IS_ADC_SEQ(SEQ) (((SEQ) != (uint32_t)0x00000000) && (((SEQ) & (uint32_t)0xFFFFF00) == ((uint32_t)0x00)))`

Макрос проверки попадания масок секвенсоров в допустимый диапазон.

- `#define IS_ADC_SEQ_IT_RATE(IT_RATE) (((IT_RATE) > ((uint32_t)0x0)) && ((IT_RATE) < ((uint32_t)0x100)))`

Проверка значения количества перезапусков модулей АЦП секвенсором после которого генерируется прерывание, на попадание в допустимый диапазон.

- `#define IS_ADC_SEQ_CONVERSION_COUNT(CONVERSION_COUNT) (((CONVERSION_COUNT) > ((uint32_t)0x0)) && ((CONVERSION_COUNT) <= ((uint32_t)0x100)))`

Проверка значения количества перезапусков модулей АЦП секвенсором после запуска секвенсора по событию на попадание в допустимый диапазон.

- `#define IS_ADC_SEQ_CONVERSION_DELAY(CONVERSION_DELAY) ((CONVERSION_DELAY) < ((uint32_t)0x1000000))`

Проверка значения задержки запуска преобразования модулем АЦП на попадание в допустимый диапазон.

- `#define IS_ADC_PHASE(PHASE) ((PHASE) < ((uint32_t)0x1000))`

Проверка значения задержки начала преобразования модулем АЦП после запуска модуля секвенсором на попадание в допустимый диапазон.

- `#define IS_ADC_DC_THRESHOLD(THRESHOLD) ((THRESHOLD) < ((uint32_t)0x1000))`

Проверка значения порога диапазона срабатывания компаратора на попадание в допустимый диапазон.

- `#define IS_ADC_SEQ_START_EVENT(START_EVENT)`

Макрос проверки аргументов типа `ADC_SEQ_StartEvent_TypeDef`.

- `#define IS_ADC_AVERAGE(AVERAGE)`

Макрос проверки аргументов типа `ADC_Average_TypeDef`.

- `#define IS_ADC_DC_CHANNEL(CHANNEL)`

Макрос проверки аргументов типа `ADC_DC_Channel_TypeDef`.

- `#define IS_ADC_DC_MODE(MODE)`

Макрос проверки аргументов типа `ADC_DC_Mode_TypeDef`.

- `#define IS_ADC_DC_CONDITION(CONDITION)`

Макрос проверки аргументов типа `ADC_DC_Condition_TypeDef`.

- `#define IS_ADC_SEQ_FIFO_LEVEL(FIFO_LEVEL)`

Макрос проверки аргументов типа `ADC_SEQ_FIFOLevel_TypeDef`.

- `#define IS_ADC_DC_MODULE(MODULE)`

Макрос проверки аргументов типа `ADC_DC_Module_TypeDef`.

- `#define IS_ADC_SEQ_MODULE(MODULE)`

Макрос проверки аргументов типа `ADC_SEQ_Module_TypeDef`.

- `#define IS_ADC_MODULE(MODULE)`

Макрос проверки аргументов типа `ADC_Module_TypeDef`.

- `#define IS_ADC_RESOLUTION(RESOLUTION)`

Макрос проверки аргументов типа `ADC_Resolution_TypeDef`.

- `#define IS_ADC_MEASURE(MEASURE)`
Макрос проверки аргументов типа `ADC_Measure_TypeDef`.
- `#define IS_ADC_MODE(MODE)`
Макрос проверки аргументов типа `ADC_Mode_TypeDef`.

Перечисления

- `enum ADC_SEQ_StartEvent_TypeDef {`
`ADC_SEQ_StartEvent_SWReq = ((uint32_t)0x0), ADC_SEQ_StartEvent_CMP0 =`
`((uint32_t)0x1), ADC_SEQ_StartEvent_CMP1 = ((uint32_t)0x2), ADC_SEQ_StartEvent_↵`
`CMP2 = ((uint32_t)0x3),`
`ADC_SEQ_StartEvent_ITGPIO = ((uint32_t)0x4), ADC_SEQ_StartEvent_TIM = ((uint32_↵`
`t)0x5), ADC_SEQ_StartEvent_PWM0 = ((uint32_t)0x6), ADC_SEQ_StartEvent_PWM1 =`
`((uint32_t)0x7),`
`ADC_SEQ_StartEvent_PWM2 = ((uint32_t)0x8), ADC_SEQ_StartEvent_PWM3 =`
`((uint32_t)0x9), ADC_SEQ_StartEvent_PWM4 = ((uint32_t)0xA), ADC_SEQ_StartEvent_↵`
`PWM5 = ((uint32_t)0xB),`
`ADC_SEQ_StartEvent_Cycle = ((uint32_t)0xF) }`
 События запуска секвенсоров.
- `enum ADC_Average_TypeDef {`
`ADC_Average_Disable, ADC_Average_2, ADC_Average_4, ADC_Average_8,`
`ADC_Average_16, ADC_Average_32, ADC_Average_64 }`
 Количество измерений, используемых для получения результата преобразования.
- `enum ADC_DC_Channel_TypeDef {`
`ADC_DC_Channel_0, ADC_DC_Channel_1, ADC_DC_Channel_2, ADC_DC_Channel_3,`
`ADC_DC_Channel_4, ADC_DC_Channel_5, ADC_DC_Channel_6, ADC_DC_Channel_7,`
`ADC_DC_Channel_8, ADC_DC_Channel_9, ADC_DC_Channel_10, ADC_DC_Channel_↵`
`_11,`
`ADC_DC_Channel_12, ADC_DC_Channel_13, ADC_DC_Channel_14, ADC_DC_↵`
`Channel_15,`
`ADC_DC_Channel_16, ADC_DC_Channel_17, ADC_DC_Channel_18, ADC_DC_↵`
`Channel_19,`
`ADC_DC_Channel_20, ADC_DC_Channel_21, ADC_DC_Channel_22, ADC_DC_↵`
`Channel_23,`
`ADC_DC_Channel_None }`
 Выбор канала, подключаемого к цифровому компаратору.
- `enum ADC_DC_Mode_TypeDef { ADC_DC_Mode_Multiple, ADC_DC_Mode_Single, AD↵`
`C_DC_Mode_MultipleHyst, ADC_DC_Mode_SingleHyst }`
 Режим срабатывания компаратора.
- `enum ADC_DC_Condition_TypeDef { ADC_DC_Condition_Low = ((uint32_t)0), ADC_D↵`
`C_Condition_Window = ((uint32_t)1), ADC_DC_Condition_High = ((uint32_t)3) }`
 Условие срабатывания компаратора.
- `enum ADC_SEQ_FIFOLevel_TypeDef {`
`ADC_SEQ_FIFOLevel_1 = ((uint32_t)1), ADC_SEQ_FIFOLevel_2 = ((uint32_t)2), ADC↵`
`_SEQ_FIFOLevel_4 = ((uint32_t)3), ADC_SEQ_FIFOLevel_8 = ((uint32_t)4),`
`ADC_SEQ_FIFOLevel_16 = ((uint32_t)5), ADC_SEQ_FIFOLevel_32 = ((uint32_t)6) }`
 Количество результатов измерений записанных в буфер секвенсора, по достижению которого вы-
 зывается DMA.
- `enum ADC_DC_Module_TypeDef {`
`ADC_DC_Module_0, ADC_DC_Module_1, ADC_DC_Module_2, ADC_DC_Module_3,`
`ADC_DC_Module_4, ADC_DC_Module_5, ADC_DC_Module_6, ADC_DC_Module_7,`
`ADC_DC_Module_8, ADC_DC_Module_9, ADC_DC_Module_10, ADC_DC_Module_11,`
`ADC_DC_Module_12, ADC_DC_Module_13, ADC_DC_Module_14, ADC_DC_Module_↵`
`15,`
`ADC_DC_Module_16, ADC_DC_Module_17, ADC_DC_Module_18, ADC_DC_Module_↵`

```
19,
ADC_DC_Module_20, ADC_DC_Module_21, ADC_DC_Module_22, ADC_DC_Module_23
}
```

Выбор модуля цифрового компаратора.

- enum ADC_SEQ_Module_TypeDef {
ADC_SEQ_Module_0, ADC_SEQ_Module_1, ADC_SEQ_Module_2, ADC_SEQ_Module_3,
ADC_SEQ_Module_4, ADC_SEQ_Module_5, ADC_SEQ_Module_6, ADC_SEQ_Module_7 }

Выбор модуля секвенсора.

- enum ADC_Module_TypeDef {
ADC_Module_0, ADC_Module_1, ADC_Module_2, ADC_Module_3,
ADC_Module_4, ADC_Module_5, ADC_Module_6, ADC_Module_7,
ADC_Module_8, ADC_Module_9, ADC_Module_10, ADC_Module_11 }

Выбор модуля АЦП.

- enum ADC_Resolution_TypeDef { ADC_Resolution_12bit, ADC_Resolution_10bit }

Выбор разрядности модуля АЦП.

- enum ADC_Measure_TypeDef { ADC_Measure_Single, ADC_Measure_Diff }

Выбор режима работы АЦП.

- enum ADC_Mode_TypeDef { ADC_Mode_Powerdown = ((uint32_t)0), ADC_Mode_StandBy = ((uint32_t)1), ADC_Mode_Active = ((uint32_t)3) }

Выбор режима работы АЦП.

Функции

- void ADC_DeInit (ADC_Module_TypeDef ADC_Module)
Устанавливает все регистры модуля АЦП значениями по умолчанию.
- void ADC_Init (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *ADC_InitStruct)
Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.
- void ADC_StructInit (ADC_Init_TypeDef *ADC_InitStruct)
Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.
- void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)
Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.
- void ADC_DC_Init (ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Init_TypeDef *ADC_DC_InitStruct)
Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры ADC_DC_InitStruct.
- void ADC_DC_StructInit (ADC_DC_Init_TypeDef *ADC_DC_InitStruct)
Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.
- void ADC_SEQ_DeInit (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Устанавливает все регистры выбранного секвенсора значениями по умолчанию.
- void ADC_SEQ_Init (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)
Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.
- void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)
Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.
- void ADC_Cmd (ADC_Module_TypeDef ADC_Module, FunctionalState State)
Включение модуля АЦП.
- void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)
Включение секвенсора.
- void ADC_SEQ_SWReq ()

- Программный запуск измерений всех разрешенных секвенсоров.
- `uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Получение результата измерений из буфера секвенсора.
- `uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.
- `uint32_t ADC_SEQ_GetFIFOLoad (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Получение количества измерений, сохраненных в буфере секвенсора.
- `FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.
- `void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс флага заполнения буфера секвенсора.
- `FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.
- `void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс флага пустоты буфера секвенсора.
- `void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`
Включение выходного триггера цифрового компаратора.
- `FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`
Проверка состояния выходного триггера компаратора.
- `void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)`
Сброс выходного триггера цифрового компаратора.
- `uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)`
Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.
- `void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)`
Конфигурирует выбранный секвенсор для работы с DMA.
- `void ADC_SEQ_DMACmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)`
Включает для выбранного секвенсора генерирование запросов DMA.
- `FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.
- `void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс статуса ошибки DMA.
- `void ADC_DC_ITCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`
Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: `ADC_DC_ITGenCmd` и `ADC_DC_ITMaskCmd`.
- `void ADC_DC_ITConfig (ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef ADC_DC_Condition)`
Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.
- `void ADC_DC_ITGenCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`
Разрешает компаратору генерировать сигнал прерывания.
- `void ADC_DC_ITMaskCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`
Маскирование сигнала прерывания цифрового компаратора.
- `FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`

- Проверка флагов немаскированных прерываний.
- `FlagStatus ADC_DC_ITMaskedStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`
- Проверка флагов маскированных прерываний.
- `void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)`
- Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.
- `void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)`
- Включение прерывания секвенсора.
- `void ADC_SEQ_ITConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)`
- Настройка вызова прерывания секвенсора.
- `uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
- Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.
- `void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
- Сброс счетчика прерываний секвенсора.
- `FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
- Проверка флагов немаскированных прерываний.
- `FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
- Проверка флагов маскированных прерываний.
- `void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
- Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.

8.3.1 Подробное описание

Файл содержит все прототипы функций для работы с АЦП, секвенсорами, цифровыми компараторами.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

10.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.3.2 Макросы

8.3.2.1 `#define ADC_Channel_0 ((uint32_t)0x00000001)`

Канал ADC 0

См. определение в файле niietcm4_adc.h строка 57

8.3.2.2 `#define ADC_Channel_1 ((uint32_t)0x00000002)`

Канал ADC 1

См. определение в файле niietcm4_adc.h строка 58

8.3.2.3 `#define ADC_Channel_10 ((uint32_t)0x00000400)`

Канал ADC 10

См. определение в файле niietcm4_adc.h строка 67

8.3.2.4 `#define ADC_Channel_11 ((uint32_t)0x00000800)`

Канал ADC 11

См. определение в файле niietcm4_adc.h строка 68

8.3.2.5 `#define ADC_Channel_12 ((uint32_t)0x00001000)`

Канал ADC 12

См. определение в файле niietcm4_adc.h строка 69

8.3.2.6 `#define ADC_Channel_13 ((uint32_t)0x00002000)`

Канал ADC 13

См. определение в файле niietcm4_adc.h строка 70

8.3.2.7 `#define ADC_Channel_14 ((uint32_t)0x00004000)`

Канал ADC 14

См. определение в файле niietcm4_adc.h строка 71

8.3.2.8 `#define ADC_Channel_15 ((uint32_t)0x00008000)`

Канал ADC 15

См. определение в файле niietcm4_adc.h строка 72

8.3.2.9 `#define ADC_Channel_16 ((uint32_t)0x00010000)`

Канал ADC 16

См. определение в файле `niietcm4_adc.h` строка 73

8.3.2.10 `#define ADC_Channel_17 ((uint32_t)0x00020000)`

Канал ADC 17

См. определение в файле `niietcm4_adc.h` строка 74

8.3.2.11 `#define ADC_Channel_18 ((uint32_t)0x00040000)`

Канал ADC 18

См. определение в файле `niietcm4_adc.h` строка 75

8.3.2.12 `#define ADC_Channel_19 ((uint32_t)0x00080000)`

Канал ADC 19

См. определение в файле `niietcm4_adc.h` строка 76

8.3.2.13 `#define ADC_Channel_2 ((uint32_t)0x00000004)`

Канал ADC 2

См. определение в файле `niietcm4_adc.h` строка 59

8.3.2.14 `#define ADC_Channel_20 ((uint32_t)0x00100000)`

Канал ADC 20

См. определение в файле `niietcm4_adc.h` строка 77

8.3.2.15 `#define ADC_Channel_21 ((uint32_t)0x00200000)`

Канал ADC 21

См. определение в файле `niietcm4_adc.h` строка 78

8.3.2.16 `#define ADC_Channel_22 ((uint32_t)0x00400000)`

Канал ADC 22

См. определение в файле `niietcm4_adc.h` строка 79

8.3.2.17 `#define ADC_Channel_23 ((uint32_t)0x00800000)`

Канал ADC 23

См. определение в файле `niietcm4_adc.h` строка 80

8.3.2.18 `#define ADC_Channel_3 ((uint32_t)0x00000008)`

Канал ADC 3

См. определение в файле niietcm4_adc.h строка 60

8.3.2.19 `#define ADC_Channel_4 ((uint32_t)0x00000010)`

Канал ADC 4

См. определение в файле niietcm4_adc.h строка 61

8.3.2.20 `#define ADC_Channel_5 ((uint32_t)0x00000020)`

Канал ADC 5

См. определение в файле niietcm4_adc.h строка 62

8.3.2.21 `#define ADC_Channel_6 ((uint32_t)0x00000040)`

Канал ADC 6

См. определение в файле niietcm4_adc.h строка 63

8.3.2.22 `#define ADC_Channel_7 ((uint32_t)0x00000080)`

Канал ADC 7

См. определение в файле niietcm4_adc.h строка 64

8.3.2.23 `#define ADC_Channel_8 ((uint32_t)0x00000100)`

Канал ADC 8

См. определение в файле niietcm4_adc.h строка 65

8.3.2.24 `#define ADC_Channel_9 ((uint32_t)0x00000200)`

Канал ADC 9

См. определение в файле niietcm4_adc.h строка 66

8.3.2.25 `#define ADC_Channel_All ((uint32_t)0x00FFFFFF)`

Все каналы

См. определение в файле niietcm4_adc.h строка 81

8.3.2.26 `#define ADC_Channel_None ((uint32_t)0x00000000)`

Канал ADC не выбран

См. определение в файле niietcm4_adc.h строка 56

Используется в ADC_SEQ_StructInit().

8.3.2.27 `#define ADC_DC_0 ((uint32_t)0x00000001)`

Цифровой компаратор 0

См. определение в файле niietcm4_adc.h строка 97

8.3.2.28 `#define ADC_DC_1 ((uint32_t)0x00000002)`

Цифровой компаратор 1

См. определение в файле `niietcm4_adc.h` строка 98

8.3.2.29 `#define ADC_DC_10 ((uint32_t)0x00000400)`

Цифровой компаратор 10

См. определение в файле `niietcm4_adc.h` строка 107

8.3.2.30 `#define ADC_DC_11 ((uint32_t)0x00000800)`

Цифровой компаратор 11

См. определение в файле `niietcm4_adc.h` строка 108

8.3.2.31 `#define ADC_DC_12 ((uint32_t)0x00001000)`

Цифровой компаратор 12

См. определение в файле `niietcm4_adc.h` строка 109

8.3.2.32 `#define ADC_DC_13 ((uint32_t)0x00002000)`

Цифровой компаратор 13

См. определение в файле `niietcm4_adc.h` строка 110

8.3.2.33 `#define ADC_DC_14 ((uint32_t)0x00004000)`

Цифровой компаратор 14

См. определение в файле `niietcm4_adc.h` строка 111

8.3.2.34 `#define ADC_DC_15 ((uint32_t)0x00008000)`

Цифровой компаратор 15

См. определение в файле `niietcm4_adc.h` строка 112

8.3.2.35 `#define ADC_DC_16 ((uint32_t)0x00010000)`

Цифровой компаратор 16

См. определение в файле `niietcm4_adc.h` строка 113

8.3.2.36 `#define ADC_DC_17 ((uint32_t)0x00020000)`

Цифровой компаратор 17

См. определение в файле `niietcm4_adc.h` строка 114

8.3.2.37 `#define ADC_DC_18 ((uint32_t)0x00040000)`

Цифровой компаратор 18

См. определение в файле niietcm4_adc.h строка 115

8.3.2.38 `#define ADC_DC_19 ((uint32_t)0x00080000)`

Цифровой компаратор 19

См. определение в файле niietcm4_adc.h строка 116

8.3.2.39 `#define ADC_DC_2 ((uint32_t)0x00000004)`

Цифровой компаратор 2

См. определение в файле niietcm4_adc.h строка 99

8.3.2.40 `#define ADC_DC_20 ((uint32_t)0x00100000)`

Цифровой компаратор 20

См. определение в файле niietcm4_adc.h строка 117

8.3.2.41 `#define ADC_DC_21 ((uint32_t)0x00200000)`

Цифровой компаратор 21

См. определение в файле niietcm4_adc.h строка 118

8.3.2.42 `#define ADC_DC_22 ((uint32_t)0x00400000)`

Цифровой компаратор 22

См. определение в файле niietcm4_adc.h строка 119

8.3.2.43 `#define ADC_DC_23 ((uint32_t)0x00800000)`

Цифровой компаратор 23

См. определение в файле niietcm4_adc.h строка 120

8.3.2.44 `#define ADC_DC_3 ((uint32_t)0x00000008)`

Цифровой компаратор 3

См. определение в файле niietcm4_adc.h строка 100

8.3.2.45 `#define ADC_DC_4 ((uint32_t)0x00000010)`

Цифровой компаратор 4

См. определение в файле niietcm4_adc.h строка 101

8.3.2.46 `#define ADC_DC_5 ((uint32_t)0x00000020)`

Цифровой компаратор 5

См. определение в файле niietcm4_adc.h строка 102

8.3.2.47 `#define ADC_DC_6 ((uint32_t)0x00000040)`

Цифровой компаратор 6

См. определение в файле `niietcm4_adc.h` строка 103

8.3.2.48 `#define ADC_DC_7 ((uint32_t)0x00000080)`

Цифровой компаратор 7

См. определение в файле `niietcm4_adc.h` строка 104

8.3.2.49 `#define ADC_DC_8 ((uint32_t)0x00000100)`

Цифровой компаратор 8

См. определение в файле `niietcm4_adc.h` строка 105

8.3.2.50 `#define ADC_DC_9 ((uint32_t)0x00000200)`

Цифровой компаратор 9

См. определение в файле `niietcm4_adc.h` строка 106

8.3.2.51 `#define ADC_DC_All ((uint32_t)0x00FFFFFF)`

Все цифровые компараторы

См. определение в файле `niietcm4_adc.h` строка 121

8.3.2.52 `#define ADC_DC_None ((uint32_t)0x00000000)`

Цифровой компаратор не выбран

См. определение в файле `niietcm4_adc.h` строка 96

Используется в `ADC_SEQ_StructInit()`.

8.3.2.53 `#define ADC_SEQ_0 ((uint32_t)0x00000001)`

Секвенсор 0

См. определение в файле `niietcm4_adc.h` строка 137

8.3.2.54 `#define ADC_SEQ_1 ((uint32_t)0x00000002)`

Секвенсор 1

См. определение в файле `niietcm4_adc.h` строка 138

8.3.2.55 `#define ADC_SEQ_2 ((uint32_t)0x00000004)`

Секвенсор 2

См. определение в файле `niietcm4_adc.h` строка 139

8.3.2.56 `#define ADC_SEQ_3 ((uint32_t)0x00000008)`

Секвенсор 3

См. определение в файле niietcm4_adc.h строка 140

8.3.2.57 `#define ADC_SEQ_4 ((uint32_t)0x00000010)`

Секвенсор 4

См. определение в файле niietcm4_adc.h строка 141

8.3.2.58 `#define ADC_SEQ_5 ((uint32_t)0x00000020)`

Секвенсор 5

См. определение в файле niietcm4_adc.h строка 142

8.3.2.59 `#define ADC_SEQ_6 ((uint32_t)0x00000040)`

Секвенсор 6

См. определение в файле niietcm4_adc.h строка 143

8.3.2.60 `#define ADC_SEQ_7 ((uint32_t)0x00000080)`

Секвенсор 7

См. определение в файле niietcm4_adc.h строка 144

8.3.2.61 `#define IS_ADC_AVERAGE(AVERAGE)`

Макроопределение:

```
((AVERAGE) == ADC_Average_Disable) || \
    ((AVERAGE) == ADC_Average_2)      || \
    ((AVERAGE) == ADC_Average_4)      || \
    ((AVERAGE) == ADC_Average_8)      || \
    ((AVERAGE) == ADC_Average_16)     || \
    ((AVERAGE) == ADC_Average_32)     || \
    ((AVERAGE) == ADC_Average_64))
```

Макрос проверки аргументов типа `ADC_Average_TypeDef`.

См. определение в файле niietcm4_adc.h строка 255

Используется в `ADC_Init()`.

8.3.2.62 `#define IS_ADC_DC_CHANNEL(CHANNEL)`

Макроопределение:

```
((CHANNEL) == ADC_DC_Channel_0) || \
    ((CHANNEL) == ADC_DC_Channel_1) || \
    ((CHANNEL) == ADC_DC_Channel_2) || \
    ((CHANNEL) == ADC_DC_Channel_3) || \
    ((CHANNEL) == ADC_DC_Channel_4) || \
    ((CHANNEL) == ADC_DC_Channel_5) || \
    ((CHANNEL) == ADC_DC_Channel_6) || \
    ((CHANNEL) == ADC_DC_Channel_7) || \
    ((CHANNEL) == ADC_DC_Channel_8) || \
    ((CHANNEL) == ADC_DC_Channel_9) || \
    ((CHANNEL) == ADC_DC_Channel_10) || \
    ((CHANNEL) == ADC_DC_Channel_11)
```

```

((CHANNEL) == ADC_DC_Channel_12) || \
((CHANNEL) == ADC_DC_Channel_13) || \
((CHANNEL) == ADC_DC_Channel_14) || \
((CHANNEL) == ADC_DC_Channel_15) || \
((CHANNEL) == ADC_DC_Channel_16) || \
((CHANNEL) == ADC_DC_Channel_17) || \
((CHANNEL) == ADC_DC_Channel_18) || \
((CHANNEL) == ADC_DC_Channel_19) || \
((CHANNEL) == ADC_DC_Channel_20) || \
((CHANNEL) == ADC_DC_Channel_21) || \
((CHANNEL) == ADC_DC_Channel_22) || \
((CHANNEL) == ADC_DC_Channel_23) || \
((CHANNEL) == ADC_DC_Channel_None))

```

Макрос проверки аргументов типа [ADC_DC_Channel_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 300

Используется в ADC_DC_Init().

8.3.2.63 `#define IS_ADC_DC_CONDITION(CONDITION)`

Макроопределение:

```

(((CONDITION) == ADC_DC_Condition_Low) || \
  ((CONDITION) == ADC_DC_Condition_Window) || \
  ((CONDITION) == ADC_DC_Condition_High))

```

Макрос проверки аргументов типа [ADC_DC_Condition_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 362

Используется в ADC_DC_Init() и ADC_DC_ITConfig().

8.3.2.64 `#define IS_ADC_DC_MODE(MODE)`

Макроопределение:

```

(((MODE) == ADC_DC_Mode_Single) || \
  ((MODE) == ADC_DC_Mode_Multiple) || \
  ((MODE) == ADC_DC_Mode_SingleHyst) || \
  ((MODE) == ADC_DC_Mode_MultipleHyst))

```

Макрос проверки аргументов типа [ADC_DC_Mode_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 342

Используется в ADC_DC_Init() и ADC_DC_ITConfig().

8.3.2.65 `#define IS_ADC_DC_MODULE(MODULE)`

Макроопределение:

```

(((MODULE) == ADC_DC_Module_0) || \
  ((MODULE) == ADC_DC_Module_1) || \
  ((MODULE) == ADC_DC_Module_2) || \
  ((MODULE) == ADC_DC_Module_3) || \
  ((MODULE) == ADC_DC_Module_4) || \
  ((MODULE) == ADC_DC_Module_5) || \
  ((MODULE) == ADC_DC_Module_6) || \
  ((MODULE) == ADC_DC_Module_7) || \
  ((MODULE) == ADC_DC_Module_8) || \
  ((MODULE) == ADC_DC_Module_9) || \
  ((MODULE) == ADC_DC_Module_10) || \
  ((MODULE) == ADC_DC_Module_11) || \
  ((MODULE) == ADC_DC_Module_12) || \
  ((MODULE) == ADC_DC_Module_13) || \
  ((MODULE) == ADC_DC_Module_14) || \
  ((MODULE) == ADC_DC_Module_15))

```



```
((MODULE) == ADC_DC_Module_16) || \
((MODULE) == ADC_DC_Module_17) || \
((MODULE) == ADC_DC_Module_18) || \
((MODULE) == ADC_DC_Module_19) || \
((MODULE) == ADC_DC_Module_20) || \
((MODULE) == ADC_DC_Module_21) || \
((MODULE) == ADC_DC_Module_22) || \
((MODULE) == ADC_DC_Module_23))
```

Макрос проверки аргументов типа [ADC_DC_Module_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 427

Используется в ADC_DC_Cmd(), ADC_DC_DeInit(), ADC_DC_GetLastData(), ADC_DC_ITCmd(), ADC_DC_ITConfig(), ADC_DC_ITGenCmd(), ADC_DC_ITMaskCmd(), ADC_DC_ITMaskedStatus(), ADC_DC_ITRawStatus(), ADC_DC_ITStatusClear(), ADC_DC_TrigStatus() и ADC_DC_TrigStatusClear().

8.3.2.66 #define IS_ADC_MEASURE(MEASURE)

Макроопределение:

```
((MEASURE) == ADC_Measure_Single) || \
((MEASURE) == ADC_Measure_Diff))
```

Макрос проверки аргументов типа [ADC_Measure_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 549

Используется в ADC_Init().

8.3.2.67 #define IS_ADC_MODE(MODE)

Макроопределение:

```
((MODE) == ADC_Mode_Powerdown) || \
((MODE) == ADC_Mode_StandBy) || \
((MODE) == ADC_Mode_Active))
```

Макрос проверки аргументов типа [ADC_Mode_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 567

Используется в ADC_Init().

8.3.2.68 #define IS_ADC_MODULE(MODULE)

Макроопределение:

```
((MODULE) == ADC_Module_0) || \
((MODULE) == ADC_Module_1) || \
((MODULE) == ADC_Module_2) || \
((MODULE) == ADC_Module_3) || \
((MODULE) == ADC_Module_4) || \
((MODULE) == ADC_Module_5) || \
((MODULE) == ADC_Module_6) || \
((MODULE) == ADC_Module_7) || \
((MODULE) == ADC_Module_8) || \
((MODULE) == ADC_Module_9) || \
((MODULE) == ADC_Module_10) || \
((MODULE) == ADC_Module_11))
```

Макрос проверки аргументов типа [ADC_Module_TypeDef](#).

См. определение в файле niietcm4_adc.h строка 505

Используется в ADC_Cmd(), ADC_DeInit() и ADC_Init().

8.3.2.69 `#define IS_ADC_RESOLUTION(RESOLUTION)`

Макроопределение:

```
((RESOLUTION) == ADC_Resolution_12bit) || \
((RESOLUTION) == ADC_Resolution_10bit))
```

Макрос проверки аргументов типа `ADC_Resolution_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 532

Используется в `ADC_Init()`.

8.3.2.70 `#define IS_ADC_SEQ_FIFO_LEVEL(FIFO_LEVEL)`

Макроопределение:

```
((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_1) || \
((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_2) || \
((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_4) || \
((FIFO_LEVEL) == ADC_SEQ_FIFOLevel_8) || \
((FIFO_LEVEL) == \
ADC_SEQ_FIFOLevel_16) || \
ADC_SEQ_FIFOLevel_32))
```

Макрос проверки аргументов типа `ADC_SEQ_FIFOLevel_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 384

Используется в `ADC_SEQ_DMAConfig()`.

8.3.2.71 `#define IS_ADC_SEQ_MODULE(MODULE)`

Макроопределение:

```
((MODULE) == ADC_SEQ_Module_0) || \
((MODULE) == ADC_SEQ_Module_1) || \
((MODULE) == ADC_SEQ_Module_2) || \
((MODULE) == ADC_SEQ_Module_3) || \
((MODULE) == ADC_SEQ_Module_4) || \
((MODULE) == ADC_SEQ_Module_5) || \
((MODULE) == ADC_SEQ_Module_6) || \
((MODULE) == ADC_SEQ_Module_7))
```

Макрос проверки аргументов типа `ADC_SEQ_Module_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 472

Используется в `ADC_SEQ_Cmd()`, `ADC_SEQ_DeInit()`, `ADC_SEQ_DMAMCmd()`, `ADC_SEQ_DMAConfig()`, `ADC_SEQ_DMAErrorStatus()`, `ADC_SEQ_DMAErrorStatusClear()`, `ADC_SEQ_FIFOEmptyStatus()`, `ADC_SEQ_FIFOEmptyStatusClear()`, `ADC_SEQ_FIFOFullStatus()`, `ADC_SEQ_FIFOFullStatusClear()`, `ADC_SEQ_GetConversionCount()`, `ADC_SEQ_GetFIFOData()`, `ADC_SEQ_GetFIFOLoad()`, `ADC_SEQ_GetITCount()`, `ADC_SEQ_Init()`, `ADC_SEQ_ITCmd()`, `ADC_SEQ_ITConfig()`, `ADC_SEQ_ITCountRst()`, `ADC_SEQ_ITMaskedStatus()`, `ADC_SEQ_ITRawStatus()` и `ADC_SEQ_ITStatusClear()`.

8.3.2.72 `#define IS_ADC_SEQ_START_EVENT(START_EVENT)`

Макроопределение:

```
((START_EVENT) == ADC_SEQ_StartEvent_SWReq) || \
((START_EVENT) == \
ADC_SEQ_StartEvent_CMP0) || \
((START_EVENT) ==
```

```

ADC_SEQ_StartEvent_CMP1) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_CMP2) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_ITGPIO) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_TIM) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM0) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM1) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM2) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM3) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM4) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_PWM5) || \
((START_EVENT) ==
ADC_SEQ_StartEvent_Cycle))

```

Макрос проверки аргументов типа `ADC_SEQ_StartEvent_TypeDef`.

См. определение в файле `niietcm4_adc.h` строка 222

Используется в `ADC_SEQ_Init()`.

8.3.3 Перечисления

8.3.3.1 enum ADC_Average_TypeDef

Количество измерений, используемых для получения результата преобразования.

Элементы перечислений

`ADC_Average_Disable` Усреднители не используются.
`ADC_Average_2` Усреднение по 2 измерениям.
`ADC_Average_4` Усреднение по 4 измерениям.
`ADC_Average_8` Усреднение по 8 измерениям.
`ADC_Average_16` Усреднение по 16 измерениям.
`ADC_Average_32` Усреднение по 32 измерениям.
`ADC_Average_64` Усреднение по 64 измерениям.

См. определение в файле `niietcm4_adc.h` строка 240

8.3.3.2 enum ADC_DC_Channel_TypeDef

Выбор канала, подключаемого к цифровому компаратору.

Элементы перечислений

`ADC_DC_Channel_0` Результат с канала 0 будет передан на компаратор.
`ADC_DC_Channel_1` Результат с канала 1 будет передан на компаратор.
`ADC_DC_Channel_2` Результат с канала 2 будет передан на компаратор.
`ADC_DC_Channel_3` Результат с канала 3 будет передан на компаратор.
`ADC_DC_Channel_4` Результат с канала 4 будет передан на компаратор.
`ADC_DC_Channel_5` Результат с канала 5 будет передан на компаратор.
`ADC_DC_Channel_6` Результат с канала 6 будет передан на компаратор.
`ADC_DC_Channel_7` Результат с канала 7 будет передан на компаратор.

ADC_DC_Channel_8 Результат с канала 8 будет передан на компаратор.
 ADC_DC_Channel_9 Результат с канала 9 будет передан на компаратор.
 ADC_DC_Channel_10 Результат с канала 10 будет передан на компаратор.
 ADC_DC_Channel_11 Результат с канала 11 будет передан на компаратор.
 ADC_DC_Channel_12 Результат с канала 12 будет передан на компаратор.
 ADC_DC_Channel_13 Результат с канала 13 будет передан на компаратор.
 ADC_DC_Channel_14 Результат с канала 14 будет передан на компаратор.
 ADC_DC_Channel_15 Результат с канала 15 будет передан на компаратор.
 ADC_DC_Channel_16 Результат с канала 16 будет передан на компаратор.
 ADC_DC_Channel_17 Результат с канала 17 будет передан на компаратор.
 ADC_DC_Channel_18 Результат с канала 18 будет передан на компаратор.
 ADC_DC_Channel_19 Результат с канала 19 будет передан на компаратор.
 ADC_DC_Channel_20 Результат с канала 20 будет передан на компаратор.
 ADC_DC_Channel_21 Результат с канала 21 будет передан на компаратор.
 ADC_DC_Channel_22 Результат с канала 22 будет передан на компаратор.
 ADC_DC_Channel_23 Результат с канала 23 будет передан на компаратор.
 ADC_DC_Channel_None Ни один из каналов не подключен к компаратору.

См. определение в файле niietcm4_adc.h строка 267

8.3.3.3 enum ADC_DC_Condition_TypeDef

Условие срабатывания компаратора.

Элементы перечислений

ADC_DC_Condition_Low Результат меньше либо равен нижней границе.
 ADC_DC_Condition_Window Результат внутри диапазона, задаваемого границами, либо равен одной из них.
 ADC_DC_Condition_High Результат выше либо равен верхней границе.

См. определение в файле niietcm4_adc.h строка 351

8.3.3.4 enum ADC_DC_Mode_TypeDef

Режим срабатывания компаратора.

Элементы перечислений

ADC_DC_Mode_Multiple Многократный.
 ADC_DC_Mode_Single Однократный.
 ADC_DC_Mode_MultipleHyst Многократный с гистерезисом.
 ADC_DC_Mode_SingleHyst Однократный с гистерезисом.

См. определение в файле niietcm4_adc.h строка 330

8.3.3.5 enum ADC_DC_Module_TypeDef

Выбор модуля цифрового компаратора.

Элементы перечислений

ADC_DC_Module_0	Модуль цифрового компаратора 0.
ADC_DC_Module_1	Модуль цифрового компаратора 1
ADC_DC_Module_2	Модуль цифрового компаратора 2
ADC_DC_Module_3	Модуль цифрового компаратора 3
ADC_DC_Module_4	Модуль цифрового компаратора 4
ADC_DC_Module_5	Модуль цифрового компаратора 5
ADC_DC_Module_6	Модуль цифрового компаратора 6
ADC_DC_Module_7	Модуль цифрового компаратора 7
ADC_DC_Module_8	Модуль цифрового компаратора 8
ADC_DC_Module_9	Модуль цифрового компаратора 9
ADC_DC_Module_10	Модуль цифрового компаратора 10
ADC_DC_Module_11	Модуль цифрового компаратора 11
ADC_DC_Module_12	Модуль цифрового компаратора 12
ADC_DC_Module_13	Модуль цифрового компаратора 13
ADC_DC_Module_14	Модуль цифрового компаратора 14
ADC_DC_Module_15	Модуль цифрового компаратора 15
ADC_DC_Module_16	Модуль цифрового компаратора 16
ADC_DC_Module_17	Модуль цифрового компаратора 17
ADC_DC_Module_18	Модуль цифрового компаратора 18
ADC_DC_Module_19	Модуль цифрового компаратора 19
ADC_DC_Module_20	Модуль цифрового компаратора 20
ADC_DC_Module_21	Модуль цифрового компаратора 21
ADC_DC_Module_22	Модуль цифрового компаратора 22
ADC_DC_Module_23	Модуль цифрового компаратора 23

См. определение в файле niietcm4_adc.h строка 395

8.3.3.6 enum ADC_Measure_TypeDef

Выбор режима работы АЦП.

Элементы перечислений

ADC_Measure_Single	Однополярный режим измерения по каналу.
ADC_Measure_Diff	Дифференциальный режим с противоположным каналом.

См. определение в файле niietcm4_adc.h строка 539

8.3.3.7 enum ADC_Mode_TypeDef

Выбор режима работы АЦП.

Элементы перечислений

ADC_Mode_Powerdown	Модуль выключен.
ADC_Mode_StandBy	Режим ожидания.
ADC_Mode_Active	Модуль включен.

См. определение в файле niietcm4_adc.h строка 556

8.3.3.8 enum ADC_Module_TypeDef

Выбор модуля АЦП.

Элементы перечислений

ADC_Module_0 Модуль АЦП 0.
ADC_Module_1 Модуль АЦП 1
ADC_Module_2 Модуль АЦП 2
ADC_Module_3 Модуль АЦП 3
ADC_Module_4 Модуль АЦП 4
ADC_Module_5 Модуль АЦП 5
ADC_Module_6 Модуль АЦП 6
ADC_Module_7 Модуль АЦП 7
ADC_Module_8 Модуль АЦП 8
ADC_Module_9 Модуль АЦП 9
ADC_Module_10 Модуль АЦП 10
ADC_Module_11 Модуль АЦП 11

См. определение в файле niietcm4_adc.h строка 485

8.3.3.9 enum ADC_Resolution_TypeDef

Выбор разрядности модуля АЦП.

Элементы перечислений

ADC_Resolution_12bit Разрядность модуля 12 бит.
ADC_Resolution_10bit Разрядность модуля 10 бит

См. определение в файле niietcm4_adc.h строка 522

8.3.3.10 enum ADC_SEQ_FIFOLevel_TypeDef

Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA.

Элементы перечислений

ADC_SEQ_FIFOLevel_1 Запрос DMA после заполнения 1 ячейки в буфере.
ADC_SEQ_FIFOLevel_2 Запрос DMA после заполнения 2 ячеек в буфере.
ADC_SEQ_FIFOLevel_4 Запрос DMA после заполнения 4 ячеек в буфере.
ADC_SEQ_FIFOLevel_8 Запрос DMA после заполнения 8 ячеек в буфере.
ADC_SEQ_FIFOLevel_16 Запрос DMA после заполнения 16 ячеек в буфере.
ADC_SEQ_FIFOLevel_32 Запрос DMA после заполнения 32 ячеек в буфере.

См. определение в файле niietcm4_adc.h строка 370

8.3.3.11 enum ADC_SEQ_Module_TypeDef

Выбор модуля секвенсора.

Элементы перечислений

ADC_SEQ_Module_0 Севенсор 0.
ADC_SEQ_Module_1 Севенсор 1
ADC_SEQ_Module_2 Севенсор 2
ADC_SEQ_Module_3 Севенсор 3
ADC_SEQ_Module_4 Севенсор 4
ADC_SEQ_Module_5 Севенсор 5
ADC_SEQ_Module_6 Севенсор 6
ADC_SEQ_Module_7 Севенсор 7

См. определение в файле niietcm4_adc.h строка 456

8.3.3.12 enum ADC_SEQ_StartEvent_TypeDef

События запуска секвенсоров.

Элементы перечислений

ADC_SEQ_StartEvent_SWReq Запуск по программному запросу.
ADC_SEQ_StartEvent_CMP0 Сигнал от блока аналогового компаратора 0.
ADC_SEQ_StartEvent_CMP1 Сигнал от блока аналогового компаратора 1.
ADC_SEQ_StartEvent_CMP2 Сигнал от блока аналогового компаратора 2.
ADC_SEQ_StartEvent_ITGPIO Любое прерывание GPIO.
ADC_SEQ_StartEvent_TIM Сигнал от блока таймеров.
ADC_SEQ_StartEvent_PWM0 Сигнал от блока 0 ШИМ.
ADC_SEQ_StartEvent_PWM1 Сигнал от блока 1 ШИМ.
ADC_SEQ_StartEvent_PWM2 Сигнал от блока 2 ШИМ.
ADC_SEQ_StartEvent_PWM3 Сигнал от блока 3 ШИМ.
ADC_SEQ_StartEvent_PWM4 Сигнал от блока 4 ШИМ.
ADC_SEQ_StartEvent_PWM5 Сигнал от блока 5 ШИМ.
ADC_SEQ_StartEvent_Cycle Циклическая работа сразу после запуска секвенсора

См. определение в файле niietcm4_adc.h строка 201

8.3.4 Функции

8.3.4.1 void ADC_Cmd (ADC_Module_TypeDef ADC_Module, FunctionalState State)

Включение модуля АЦП.

Аргументы

ADC_Module	Выбор АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 108

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1017

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.3 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

ADC_DC_↔ Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 300

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.4 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 1033

Перекрестные ссылки IS_ADC_DC_MODULE.


```
8.3.4.5 void ADC_DC_Init ( ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Init_TypeDef * ADC_DC_InitStruct )
```

Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры `ADC_DC_InitStruct`.

Аргументы

<code>ADC_DC_↵ Module</code>	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
<code>ADC_DC_↵ InitStruct</code>	Указатель на структуру типа ADC_DC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 319

Перекрестные ссылки `ADC_DC_Init_TypeDef::ADC_DC_Channel`, `ADC_DC_Init_TypeDef::ADC_DC_Condition`, `ADC_DC_Init_TypeDef::ADC_DC_Mode`, `ADC_DC_Init_TypeDef::ADC_DC_ThresholdHigh`, `ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow`, `IS_ADC_DC`, `IS_ADC_DC_CHANNEL`, `IS_ADC_DC_CONDITION`, `IS_ADC_DC_MODE` и `IS_ADC_DC_THRESHOLD`.

```
8.3.4.6 void ADC_DC_ITCmd ( ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState
State )
```

Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: [ADC_DC_ITGenCmd](#) и [ADC_DC_ITMaskCmd](#).

Аргументы

<code>ADC_DC_↵ Module</code>	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 589

Перекрестные ссылки `ADC_DC_ITGenCmd()`, `ADC_DC_ITMaskCmd()`, `IS_ADC_DC_MODULE` и `IS_FUNCTIONAL_STATE`.

```
8.3.4.7 void ADC_DC_ITConfig ( ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef
ADC_DC_Condition )
```

Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.

Аргументы

<code>ADC_DC_↵ Module</code>	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
<code>ADC_DC_↵ Mode</code>	Режим срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Mode_TypeDef .
<code>ADC_DC_↵ Condition</code>	Условие срабатывания компаратора. Параметр принимает любое значение из ADC_DC_Condition_TypeDef .

Возвращаемые значения

нет	
-----	--

См. определение в файле `niietcm4_adc.c` строка 613

Перекрестные ссылки `IS_ADC_DC_CONDITION`, `IS_ADC_DC_MODE` и `IS_ADC_DC_MODULE`.

8.3.4.8 `void ADC_DC_ITGenCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

<code>ADC_DC_↔Module</code>	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет	
-----	--

См. определение в файле `niietcm4_adc.c` строка 546

Перекрестные ссылки `IS_ADC_DC_MODULE` и `IS_FUNCTIONAL_STATE`.

Используется в `ADC_DC_ITCmd()`.

8.3.4.9 `void ADC_DC_ITMaskCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

<code>ADC_DC_↔Module</code>	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет	
-----	--

См. определение в файле `niietcm4_adc.c` строка 563

Перекрестные ссылки `IS_ADC_DC_MODULE` и `IS_FUNCTIONAL_STATE`.

Используется в `ADC_DC_ITCmd()`.

8.3.4.10 `FlagStatus ADC_DC_ITMaskedStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`

Проверка флагов маскированных прерываний.

Аргументы

<code>ADC_DC_↔Module</code>	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_DC_Module_TypeDef .
-----------------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 655

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.11 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔ C_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 630

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.12 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔ C_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 681

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.13 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

ADC_DC_↔ InitStruct	Указатель на структуру типа ADC_DC_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 342

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Channel_None, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Condition_Low, ADC_DC_Init_↔
__TypeDef::ADC_DC_Mode, ADC_DC_Mode_Single, ADC_DC_Init_TypeDef::ADC_DC_↔
ThresholdHigh и ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow.

8.3.4.14 FlagStatus ADC_DC_TriggerStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из ADC_DC_↔ Module_TypeDef .
--------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние триггера.
------------	-----------------------------

См. определение в файле niietcm4_adc.c строка 1051

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.15 void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Сброс выходного триггера цифрового компаратора.

Внимание

Одновременно со сбросом триггеров 0 и 1 компаратора сбрасываются триггеры 10 и 11 компаратора соответственно. То же самое справедливо и для обратного случая. Это происходит аппаратно и программными методами не обходится.

Аргументы

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из ADC_↔ DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1079

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

8.3.4.16 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_↔ TypeDef .
------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 123

Перекрестные ссылки ADC_Module_0, ADC_Module_1, ADC_Module_10, ADC_Module_11, ADC_Module_2, ADC_Module_3, ADC_Module_4, ADC_Module_5, ADC_Module_6, ADC_Module_7, ADC_Module_8, ADC_Module_9 и IS_ADC_MODULE.

8.3.4.17 void ADC_Init (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *
ADC_InitStruct)

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_TypeDef .
ADC_InitStruct	Указатель на структуру типа ADC_Init_TypeDef , которая содержит конфигурационную информацию.

См. определение в файле niietcm4_adc.c строка 199

Перекрестные ссылки [ADC_Init_TypeDef::ADC_Average](#), [ADC_Init_TypeDef::ADC_Measure_A](#), [ADC_Init_TypeDef::ADC_Measure_B](#), [ADC_Init_TypeDef::ADC_Mode](#), [ADC_Module_0](#), [ADC_Module_1](#), [ADC_Module_10](#), [ADC_Module_11](#), [ADC_Module_2](#), [ADC_Module_3](#), [ADC_Module_4](#), [ADC_Module_5](#), [ADC_Module_6](#), [ADC_Module_7](#), [ADC_Module_8](#), [ADC_Module_9](#), [ADC_Init_TypeDef::ADC_Phase](#), [ADC_Init_TypeDef::ADC_Resolution](#), [IS_ADC_AVERAGE](#), [IS_ADC_MEASURE](#), [IS_ADC_MODE](#), [IS_ADC_MODULE](#), [IS_ADC_PHASE](#) и [IS_ADC_RESOLUTION](#).

```
8.3.4.18 void ADC_SEQ_Cmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                          FunctionalState State )
```

Включение секвенсора.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 848

Перекрестные ссылки [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

```
8.3.4.19 void ADC_SEQ_DeInit ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module )
```

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

ADC_SEQ_Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 358

Перекрестные ссылки [ADC_SEQ_Module_0](#), [ADC_SEQ_Module_1](#), [ADC_SEQ_Module_2](#), [ADC_SEQ_Module_3](#), [ADC_SEQ_Module_4](#), [ADC_SEQ_Module_5](#), [ADC_SEQ_Module_6](#), [ADC_SEQ_Module_7](#) и [IS_ADC_SEQ_MODULE](#).

```
8.3.4.20 void ADC_SEQ_DMAMCmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                              FunctionalState State )
```

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 489

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.21 void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из ADC_SEQ_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 472

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

8.3.4.22 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущие состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 505

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.23 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 530

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.24 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 976

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.25 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1001

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.26 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 936

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.27 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 961

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.28 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 898

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.29 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение результата измерений из буфера секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 880

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
8.3.4.30 uint32_t ADC_SEQ_GetFIFOLoad ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)
```

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

Data	Результат измерения.
------	----------------------

См. определение в файле niietcm4_adc.c строка 916

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.31 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

ITCount	
---------	--

См. определение в файле niietcm4_adc.c строка 750

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.32 void ADC_SEQ_Init (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 418

Перекрестные ссылки ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent, ADC_SEQ_Init_TypeDef::ADC_SEQ_SWReqEn, IS_ADC_CHANNEL, IS_ADC_DC, IS_ADC_SEQ_CONVERSION_COUNT, IS_ADC_SEQ_CONVERSION_DELAY, IS_ADC_SEQ_MODULE, IS_ADC_SEQ_START_EVENT и IS_FUNCTIONAL_STATE.

8.3.4.33 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
FunctionalState State)

Включение прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 697

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.34 void ADC_SEQ_ITConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)

Настройка вызова прерывания секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ ITRate	Значение количества перезапусков модулей АЦП секвенсором после которого генерируется прерывание. Параметр принимает любое значение из диапазона 1 - 256.
ADC_SEQ_↔ ITCountSEQRst	Разрешение сброса счетчика прерываний по запуску секвенсора. Если запретить, то счетчик можно будет сбрасывать только программно через ADC_SEQ_IT↔ CountRst . Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 725

Перекрестные ссылки IS_ADC_SEQ_IT_RATE, IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.35 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 768

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.36 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 807

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.37 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов немаскированных прерываний.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

FlagStatus	Текущее состояние флага.
------------	--------------------------

См. определение в файле niietcm4_adc.c строка 782

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.38 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.

Аргументы

ADC_SEQ_↔ Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_↔ Module_TypeDef .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 832

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.39 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которую необходимо проинициализировать.
-------------------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 453

Перекрестные ссылки `ADC_Channel_None`, `ADC_SEQ_Init_TypeDef::ADC_Channels`, `ADC_D↔C_None`, `ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount`, `ADC_SEQ_Init_TypeDef::AD↔C_SEQ_ConversionDelay`, `ADC_SEQ_Init_TypeDef::ADC_SEQ_DC`, `ADC_SEQ_Init_TypeDef↔::ADC_SEQ_StartEvent`, `ADC_SEQ_StartEvent_SWReq` и `ADC_SEQ_Init_TypeDef::ADC_SE↔Q_SWReqEn`.

8.3.4.40 `void ADC_SEQ_SWReq ()`

Программный запуск измерений всех разрешенных секвенсоров.

Возвращаемые значения

нет

См. определение в файле `niietcm4_adc.c` строка 868

8.3.4.41 `void ADC_StructInit (ADC_Init_TypeDef * ADC_InitStruct)`

Заполнение каждого члена структуры `ADC_InitStruct` значениями по умолчанию.

Аргументы

<code>ADC_Init↔Struct</code>	Указатель на структуру типа <code>ADC_Init_TypeDef</code> , которую необходимо проинициализировать.
------------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_adc.c` строка 283

Перекрестные ссылки `ADC_Init_TypeDef::ADC_Average`, `ADC_Average_Disable`, `ADC_Init_↔TypeDef::ADC_Measure_A`, `ADC_Init_TypeDef::ADC_Measure_B`, `ADC_Measure_Single`, `ADC↔_Init_TypeDef::ADC_Mode`, `ADC_Mode_Powerdown`, `ADC_Init_TypeDef::ADC_Phase`, `ADC_↔Init_TypeDef::ADC_Resolution` и `ADC_Resolution_12bit`.

8.4 Файл `niietcm4_bootflash.c`

Файл содержит реализацию всех функции для работы с загрузочной флеш.

```
#include "niietcm4_bootflash.h"
```

Функции

- `void BOOTFLASH_Init (uint32_t SysClkFreq)`
Инициализирует тайминги доступа для контроллера загрузочной флеш.
- `BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()`
Статус работы контроллера загрузочной флеш.
- `void BOOTFLASH_OperationStatusClear ()`
Очищает статус работы контроллера загрузочной флеш.
- `void BOOTFLASH_FullErase ()`
Полная очистка основной области загрузочной флеш.
- `void BOOTFLASH_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)`
Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.
- `void BOOTFLASH_PageErase (uint32_t PageNum)`

- Стирание указанной страницы основной области загрузочной флеш.
- void **BOOTFLASH_Info_Write** (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.
- void **BOOTFLASH_Info_PageErase** (uint32_t PageNum)
Стирание указанной страницы информационной области загрузочной флеш.
- void **BOOTFLASH_ITCmd** (**FunctionalState** State)
Включение прерывания по завершению чтения/записи/стирания.

8.4.1 Подробное описание

Файл содержит реализацию всех функции для работы с загрузочной флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

07.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.4.2 Функции

8.4.2.1 void **BOOTFLASH_FullErase** ()

Полная очистка основной области загрузочной флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 112

Перекрестные ссылки **BOOTFLASH_MAGIC_KEY**.

8.4.2.2 void BOOTFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области загрузочной флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 179

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_↔
BOOTFLASH_INFO_PAGE_NUM.

8.4.2.3 void BOOTFLASH_Info_Write (uint32_t Address, uint32_t Data0, uint32_t Data1,
uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 163

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.4.2.4 void BOOTFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера загрузочной флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
------------	---------------------------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 75

8.4.2.5 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.4.2.6 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из BOOTFLASH_Status_TypeDef .
--------	---

См. определение в файле niietcm4_bootflash.c строка 88

8.4.2.7 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.

См. определение в файле niietcm4_bootflash.c строка 102

8.4.2.8 void BOOTFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области загрузочной флэш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 144

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_BOOTFLASH_PAGE_NUM.

8.4.2.9 void BOOTFLASH_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в основную область загрузочной флэш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 128

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.5 Файл niietcm4_bootflash.h

Файл содержит все прототипы функций для загрузочной флэш.

```
#include "niietcm4.h"
```

Макросы

- `#define BOOTFLASH_MAGIC_KEY ((uint32_t)0xA4420000)`
Ключ для проведения операций с контроллером загрузочной флеш.
- `#define BOOTFLASH_PAGE_SIZE_BYTES ((uint32_t)8192)`
- `#define BOOTFLASH_PAGE_TOTAL ((uint32_t)128)`
- `#define BOOTFLASH_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BOOTFLASH_PAGE_TOTAL)`
- `#define IS_BOOTFLASH_PAGE_NUM(PAGE_NUM) (PAGE_NUM < BOOTFLASH_PAGE_TOTAL)`
Макрос проверки номера страницы основной области загрузочной флеш на попадание в допустимый диапазон.
- `#define BOOTFLASH_INFO_PAGE_SIZE_BYTES BOOTFLASH_PAGE_SIZE_BYTES`
- `#define BOOTFLASH_INFO_PAGE_TOTAL ((uint32_t)1)`
- `#define BOOTFLASH_INFO_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BOOTFLASH_PAGE_TOTAL)`
- `#define IS_BOOTFLASH_INFO_PAGE_NUM(PAGE_NUM) (PAGE_NUM < BOOTFLASH_INFO_PAGE_TOTAL)`
Макрос проверки номера страницы информационной области загрузочной флеш на попадание в допустимый диапазон.
- `#define IS_BOOTFLASH_STATUS(STATUS)`
Макрос проверки аргументов типа `BOOTFLASH_Status_TypeDef`.

Перечисления

- `enum BOOTFLASH_Status_TypeDef { BOOTFLASH_Status_None = ((uint32_t)0), BOOTFLASH_Status_Complete = ((uint32_t)1), BOOTFLASH_Status_Error = ((uint32_t)3) }`
Статус работы контроллера загрузочной флеш-памяти.

Функции

- `void BOOTFLASH_Init (uint32_t SysClkFreq)`
Инициализирует тайминги доступа для контроллера загрузочной флеш.
- `BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()`
Статус работы контроллера загрузочной флеш.
- `void BOOTFLASH_OperationStatusClear ()`
Очищает статус работы контроллера загрузочной флеш.
- `void BOOTFLASH_ITCmd (FunctionalState State)`
Включение прерывания по завершению чтения/записи/стирания.
- `void BOOTFLASH_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)`
Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.
- `void BOOTFLASH_PageErase (uint32_t PageNum)`
Стирание указанной страницы основной области загрузочной флеш.
- `void BOOTFLASH_FullErase ()`
Полная очистка основной области загрузочной флеш.
- `void BOOTFLASH_Info_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)`
Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.

- void `BOOTFLASH_Info_PageErase` (uint32_t PageNum)

Стирание указанной страницы информационной области загрузочной флеш.

8.5.1 Подробное описание

Файл содержит все прототипы функций для загрузочной флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

18.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.5.2 Макросы

8.5.2.1 `#define BOOTFLASH_INFO_PAGE_SIZE_BYTES BOOTFLASH_PAGE_SIZE_BYTES`

Размер страницы в байтах.

См. определение в файле `niietcm4_bootflash.h` строка 80

8.5.2.2 `#define BOOTFLASH_INFO_PAGE_TOTAL ((uint32_t)1)`

Общее количество страниц.

См. определение в файле `niietcm4_bootflash.h` строка 81

8.5.2.3 `#define BOOTFLASH_INFO_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTE * BOOTFLASH_PAGE_TOTAL)`

Общий размер информационной области.

См. определение в файле `niietcm4_bootflash.h` строка 82

8.5.2.4 `#define BOOTFLASH_PAGE_SIZE_BYTES ((uint32_t)8192)`

Размер страницы в байтах.

См. определение в файле niietcm4_bootflash.h строка 62

Используется в `BOOTFLASH_Info_PageErase()` и `BOOTFLASH_PageErase()`.

8.5.2.5 `#define BOOTFLASH_PAGE_TOTAL ((uint32_t)128)`

Общее количество страниц.

См. определение в файле niietcm4_bootflash.h строка 63

8.5.2.6 `#define BOOTFLASH_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BOOTFLASH_PAGE_TOTAL)`

Общий размер основной области.

См. определение в файле niietcm4_bootflash.h строка 64

8.5.2.7 `#define IS_BOOTFLASH_STATUS(STATUS)`

Макроопределение:

```
((STATUS) == BOOTFLASH_Status_None) || \
((STATUS) == BOOTFLASH_Status_Complete) || \
((STATUS) == BOOTFLASH_Status_Error))
```

Макрос проверки аргументов типа `BOOTFLASH_Status_TypeDef`.

См. определение в файле niietcm4_bootflash.h строка 117

8.5.3 Перечисления

8.5.3.1 `enum BOOTFLASH_Status_TypeDef`

Статус работы контроллера загрузочной флеш-памяти.

Элементы перечислений

`BOOTFLASH_Status_None` Операция выполняется или отсутствует.

`BOOTFLASH_Status_Complete` Операция успешно завершена.

`BOOTFLASH_Status_Error` Операция завершена с ошибкой.

См. определение в файле niietcm4_bootflash.h строка 106

8.5.4 Функции

8.5.4.1 `void BOOTFLASH_FullErase ()`

Полная очистка основной области загрузочной флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 112

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.5.4.2 void BOOTFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области загрузочной флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 179

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_↔
BOOTFLASH_INFO_PAGE_NUM.

8.5.4.3 void BOOTFLASH_Info_Write (uint32_t Address, uint32_t Data0, uint32_t Data1,
uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 163

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.5.4.4 void BOOTFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера загрузочной флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
------------	---------------------------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 75

8.5.4.5 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.5.4.6 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из BOOTFLASH_Status_TypeDef .
--------	---

См. определение в файле niietcm4_bootflash.c строка 88

8.5.4.7 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.

См. определение в файле niietcm4_bootflash.c строка 102

8.5.4.8 void BOOTFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области загрузочной флэш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 144

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_BOOTFLASH_PAGE_NUM.

8.5.4.9 void BOOTFLASH_Write (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)

Запись 128 бит информации в основную область загрузочной флэш, начиная с указанного адреса.

Аргументы

Address	Стартовый адрес.
Data0	Нулевое (младшее) 32-битное слово данных.
Data1	Первое 32-битное слово данных.
Data2	Второе 32-битное слово данных.
Data3	Третье (старшее) 32-битное слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 128

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.6 Файл niietcm4_cap.c

Файл содержит реализацию всех функции для работы с блоками захвата

```
#include "niietcm4_cap.h"
```

Функции

- void [CAP_DeInit](#) (NT_CAP_TypeDef *CAPx)
Устанавливает все регистры блока захвата значениями по умолчанию.
- void [CAP_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_Init_TypeDef](#) *CAP_InitStruct)
Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.
- void [CAP_SyncCmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение синхронизации.
- void [CAP_StructInit](#) ([CAP_Init_TypeDef](#) *CAP_InitStruct)
Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.
- void [CAP_TimerCmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)
Разрешение работы таймера, выбранного блока захвата.
- void [CAP_SetTimer](#) (NT_CAP_TypeDef *CAPx, uint32_t TimerVal)
Установка текущего значения счетчика напрямую.
- void [CAP_SetShadowTimer](#) (NT_CAP_TypeDef *CAPx, uint32_t TimerVal)
Установка теневого значения таймера для отложенной записи.
- uint32_t [CAP_GetTimer](#) (NT_CAP_TypeDef *CAPx)
Получение текущего значения таймера.
- uint32_t [CAP_GetShadowTimer](#) (NT_CAP_TypeDef *CAPx)
Получение отложенного значения таймера.
- void [CAP_SwSync](#) (NT_CAP_TypeDef *CAPx)
Проведение программной синхронизации.
- void [CAP_PWM_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_PWM_Init_TypeDef](#) *CAP_PWM_InitStruct)
Инициализирует режим ШИМ блока CAPx согласно параметрам структуры CAP_PWM_InitStruct.
- void [CAP_PWM_StructInit](#) ([CAP_PWM_Init_TypeDef](#) *CAP_PWM_InitStruct)
Заполнение каждого члена структуры CAP_PWM_InitStruct значениями по умолчанию.
- void [CAP_PWM_SetPeriod](#) (NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)
Установка значения периода ШИМ.
- void [CAP_PWM_SetCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)
Установка значения сравнения ШИМ.
- void [CAP_PWM_SetShadowPeriod](#) (NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)
Установка значения периода ШИМ для отложенной записи.
- void [CAP_PWM_SetShadowCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)
Установка значения сравнения ШИМ для отложенной записи.
- uint32_t [CAP_PWM_GetPeriod](#) (NT_CAP_TypeDef *CAPx)
Получение текущего периода ШИМ.

- `uint32_t CAP_PWM_GetCompare` (`NT_CAP_TypeDef *CAPx`)
Получение текущего значения сравнения ШИМ.
- `uint32_t CAP_PWM_GetShadowPeriod` (`NT_CAP_TypeDef *CAPx`)
Получение отложенного значения периода ШИМ.
- `uint32_t CAP_PWM_GetShadowCompare` (`NT_CAP_TypeDef *CAPx`)
Получение отложенного значения сравнения ШИМ.
- `void CAP_Capture_Init` (`NT_CAP_TypeDef *CAPx`, `CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct`)
Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.
- `void CAP_Capture_StructInit` (`CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct`)
Заполнение каждого члена структуры CAP_Capture_InitStruct значениями по умолчанию.
- `void CAP_Capture_Cmd` (`NT_CAP_TypeDef *CAPx`, `FunctionalState State`)
Разрешение захвата для выбранного блока захвата.
- `void CAP_Capture_SetCap0` (`NT_CAP_TypeDef *CAPx`, `uint32_t Value`)
Установка значения регистра захвата 0.
- `void CAP_Capture_SetCap1` (`NT_CAP_TypeDef *CAPx`, `uint32_t Value`)
Установка значения регистра захвата 1.
- `void CAP_Capture_SetCap2` (`NT_CAP_TypeDef *CAPx`, `uint32_t Value`)
Установка значения регистра захвата 2.
- `void CAP_Capture_SetCap3` (`NT_CAP_TypeDef *CAPx`, `uint32_t Value`)
Установка значения регистра захвата 3.
- `uint32_t CAP_Capture_GetCap0` (`NT_CAP_TypeDef *CAPx`)
Получение текущего значения из регистра захвата 0.
- `uint32_t CAP_Capture_GetCap1` (`NT_CAP_TypeDef *CAPx`)
Получение текущего значения из регистра захвата 1.
- `uint32_t CAP_Capture_GetCap2` (`NT_CAP_TypeDef *CAPx`)
Получение текущего значения из регистра захвата 2.
- `uint32_t CAP_Capture_GetCap3` (`NT_CAP_TypeDef *CAPx`)
Получение текущего значения из регистра захвата 3.
- `void CAP_ITCmd` (`NT_CAP_TypeDef *CAPx`, `uint32_t CAP_ITSource`, `FunctionalState State`)
Разрешение работы прерывания выбранного блока захвата.
- `void CAP_ITForceCmd` (`NT_CAP_TypeDef *CAPx`, `uint32_t CAP_ITSource`)
Принудительный вызов прерывания выбранного блока захвата.
- `FlagStatus CAP_ITStatus` (`NT_CAP_TypeDef *CAPx`, `uint32_t CAP_ITSource`)
Чтение статуса флага источника прерывания выбранного блока захвата.
- `void CAP_ITStatusClear` (`NT_CAP_TypeDef *CAPx`, `uint32_t CAP_ITSource`)
Сброс флагов источников прерываний выбранного блока захвата.
- `FlagStatus CAP_ITPendStatus` (`NT_CAP_TypeDef *CAPx`)
Чтение статуса прерывания выбранного блока захвата.
- `void CAP_ITPendClear` (`NT_CAP_TypeDef *CAPx`)
Сброс флага прерывания выбранного блока захвата.

8.6.1 Подробное описание

Файл содержит реализацию всех функции для работы с блоками захвата

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

18.01.2016

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2016 ОАО "НИИЭТ"

8.6.2 Функции

8.6.2.1 void CAP_Capture_Cmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение захвата для выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 444

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

8.6.2.2 uint32_t CAP_Capture_GetCap0 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 0.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 519

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.3 uint32_t CAP_Capture_GetCap1 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 1.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 532

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.4 uint32_t CAP_Capture_GetCap2 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 2.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 545

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.5 uint32_t CAP_Capture_GetCap3 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 3.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 558

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.6 void CAP_Capture_Init (NT_CAP_TypeDef * CAPx, CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)

Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_Capture_InitStruct	Указатель на структуру типа CAP_Capture_Init_TypeDef, которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 386

Перекрестные ссылки CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0, CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent1, CAP_Capture_Init_TypeDef::CAP_

Capture_PolarityEvent2, CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3, CAP_Capture_Init_TypeDef::CAP_Capture_Prescale, CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent0, CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1, CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent2, CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3, CAP_Capture_Init_TypeDef::CAP_Capture_StopVal, CAP_Capture_Init_TypeDef::CAP_CaptureMode, IS_CAP_ALL_PERIPH, IS_CAP_CAPTURE_MODE, IS_CAP_CAPTURE_POLARITY, IS_CAP_CAPTURE_PRESCALE, IS_CAP_CAPTURE_STOP_VAL и IS_FUNCTIONAL_STATE.

8.6.2.7 void CAP_Capture_SetCap0 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 0.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 464

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.8 void CAP_Capture_SetCap1 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 1.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 478

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.9 void CAP_Capture_SetCap2 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 2.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 492

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.10 void CAP_Capture_SetCap3 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 3.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 506

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

8.6.2.11 `void CAP_Capture_StructInit (CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)`

Заполнение каждого члена структуры `CAP_Capture_InitStruct` значениями по умолчанию.

Аргументы

CAP_↵ Capture_Init↵ Struct	Указатель на структуру типа <code>CAP_Capture_Init_TypeDef</code> , которую необходимо проинициализировать.
----------------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 421

Перекрестные ссылки `CAP_Capture_Mode_Single`, `CAP_Capture_Polarity_PosEdge`, `CAP_↵
_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`, `CAP_Capture_Init_TypeDef::CAP_↵
_Capture_PolarityEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent2`, `CAP_↵
_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`, `CAP_Capture_Init_TypeDef::CAP_↵
_Capture_Prescale`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent0`, `CAP_Capture_Init_↵
_TypeDef::CAP_Capture_RstEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3`, `CAP_Capture_Init_TypeDef::CAP_↵
_Capture_StopVal` и `CAP_Capture_Init_TypeDef::CAP_CaptureMode`.

8.6.2.12 `void CAP_DeInit (NT_CAP_TypeDef * CAPx)`

Устанавливает все регистры блока захвата значениями по умолчанию.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 68

Перекрестные ссылки `IS_CAP_ALL_PERIPH`, `RCC_PeriphRst_CAP0`, `RCC_PeriphRst_CAP1`, `RCC_PeriphRst_CAP2`, `RCC_PeriphRst_CAP3`, `RCC_PeriphRst_CAP4`, `RCC_PeriphRst_CAP5` и `RCC_PeriphRstCmd()`.

8.6.2.13 `uint32_t CAP_GetShadowTimer (NT_CAP_TypeDef * CAPx)`

Получение отложенного значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 218

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.14 uint32_t CAP_GetTimer (NT_CAP_TypeDef * CAPx)

Получение текущего значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 205

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.15 void CAP_Init (NT_CAP_TypeDef * CAPx, CAP_Init_TypeDef * CAP_InitStruct)

Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 111

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Init_TypeDef::CAP_Mode, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut, IS_CAP_ALL_PERIPH, IS_CAP_HALT, IS_CAP_MODE и IS_CAP_SYNC_OUT.

8.6.2.16 void CAP_ITCmd (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource, FunctionalState State)

Разрешение работы прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_ITSource	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 575

Перекрестные ссылки IS_CAP_ALL_PERIPH, IS_CAP_IT_SOURCE и IS_FUNCTIONAL_STATE.

8.6.2.17 void CAP_ITForceCmd (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Принудительный вызов прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_ITSource	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 599

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

8.6.2.18 void CAP_ITPendClear (NT_CAP_TypeDef * CAPx)

Сброс флага прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 680

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.19 FlagStatus CAP_ITPendStatus (NT_CAP_TypeDef * CAPx)

Чтение статуса прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 656

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.20 FlagStatus CAP_ITStatus (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Чтение статуса флага источника прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 615

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE_SINGLE.

8.6.2.21 void CAP_ITStatusClear (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Сброс флагов источников прерываний выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 642

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

8.6.2.22 uint32_t CAP_PWM_GetCompare (NT_CAP_TypeDef * CAPx)

Получение текущего значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле niietcm4_cap.c строка 345

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.23 uint32_t CAP_PWM_GetPeriod (NT_CAP_TypeDef * CAPx)

Получение текущего периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле niietcm4_cap.c строка 332

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.24 uint32_t CAP_PWM_GetShadowCompare (NT_CAP_TypeDef * CAPx)

Получение отложенного значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле `niietcm4_cap.c` строка 371

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

8.6.2.25 `uint32_t CAP_PWM_GetShadowPeriod (NT_CAP_TypeDef * CAPx)`

Получение отложенного значения периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле `niietcm4_cap.c` строка 358

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

8.6.2.26 `void CAP_PWM_Init (NT_CAP_TypeDef * CAPx, CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)`

Инициализирует режим ШИМ блока CAPx согласно параметрам структуры `CAP_PWM_InitStruct`.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_PWM_InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_cap.c` строка 246

Перекрестные ссылки `CAP_PWM_Init_TypeDef::CAP_PWM_Compare`, `CAP_PWM_Init_TypeDef::CAP_PWM_Period`, `CAP_PWM_Init_TypeDef::CAP_PWM_Polarity`, `CAP_PWM_SetCompare()`, `CAP_PWM_SetPeriod()`, `IS_CAP_ALL_PERIPH` и `IS_CAP_PWM_POLARITY`.

8.6.2.27 `void CAP_PWM_SetCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)`

Установка значения сравнения ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 291

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

8.6.2.28 void CAP_PWM_SetPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 277

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

8.6.2.29 void CAP_PWM_SetShadowCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 319

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.30 void CAP_PWM_SetShadowPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 305

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.31 void CAP_PWM_StructInit (CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)

Заполнение каждого члена структуры CAP_PWM_InitStruct значениями по умолчанию.

Аргументы

CAP_PWM_↔ _InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef , которую необходимо проинициализировать.
--------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 263

Перекрестные ссылки CAP_PWM_Init_TypeDef::CAP_PWM_Compare, CAP_PWM_Init_↔
_TypeDef::CAP_PWM_Period, CAP_PWM_Init_TypeDef::CAP_PWM_Polarity и CAP_PWM_↔
Polarity_Pos.

8.6.2.32 void CAP_SetShadowTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)

Установка теневого значения таймера для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 192

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.33 void CAP_SetTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)

Установка текущего значения счетчика напрямую.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 178

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.34 void CAP_StructInit (CAP_Init_TypeDef * CAP_InitStruct)

Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.

Аргументы

CAP_Init_↔ Struct	Указатель на структуру типа CAP_Init_TypeDef , которую необходимо проинициализировать.
----------------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 147

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Halt_Stop, CAP_Init_TypeDef::CAP_Mode, CAP_Mode_Capture, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut и CAP_SyncOut_Bypass.

8.6.2.35 void CAP_SwSync (NT_CAP_TypeDef * CAPx)

Проведение программной синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 231

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.6.2.36 void CAP_SyncCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 132

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

8.6.2.37 void CAP_TimerCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение работы таймера, выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 163

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

8.7 Файл niietcm4_cap.h

Файл содержит все прототипы функций для блоков захвата


```
#include "niietcm4.h"
```

Структуры данных

- struct `CAP_Init_TypeDef`
Структура инициализации блока захвата в целом.
- struct `CAP_Capture_Init_TypeDef`
Структура инициализации режима захвата.
- struct `CAP_PWM_Init_TypeDef`
Структура инициализации режима ШИМ.

Макросы

- `#define IS_CAP_CAPTURE_POLARITY(CAPTURE_POLARITY)`
Макрос проверки аргументов типа `CAP_Capture_Polarity_TypeDef`.
- `#define IS_CAP_HALT(HALT)`
Макрос проверки аргументов типа `CAP_Halt_TypeDef`.
- `#define IS_CAP_SYNC_OUT(SYNC_OUT)`
Макрос проверки аргументов типа `CAP_SyncOut_TypeDef`.
- `#define IS_CAP_CAPTURE_MODE(CAPTURE_MODE)`
Макрос проверки аргументов типа `CAP_Capture_Mode_TypeDef`.
- `#define IS_CAP_PWM_POLARITY(PWM_POLARITY)`
Макрос проверки аргументов типа `CAP_PWM_Polarity_TypeDef`.
- `#define IS_CAP_MODE(MODE)`
Макрос проверки аргументов типа `CAP_Mode_TypeDef`.
- `#define IS_CAP_CAPTURE_PRESCALE(PRESCALE) ((PRESCALE) < ((uint32_t)0x40))`
Проверка значения предварительного делителя событий на попадание в допустимый диапазон.
- `#define IS_CAP_CAPTURE_STOP_VAL(STOP_VAL) ((STOP_VAL) < ((uint32_t)4))`
Проверка значения счетчика событий для остановки одиночного режима захвата на попадание в допустимый диапазон.
- `#define CAP_ITSource_GeneralInt ((uint32_t)0x01)`
- `#define CAP_ITSource_CapEvent0 ((uint32_t)0x02)`
- `#define CAP_ITSource_CapEvent1 ((uint32_t)0x04)`
- `#define CAP_ITSource_CapEvent2 ((uint32_t)0x08)`
- `#define CAP_ITSource_CapEvent3 ((uint32_t)0x10)`
- `#define CAP_ITSource_TimerOvf ((uint32_t)0x20)`
- `#define CAP_ITSource_TimerEqPeriod ((uint32_t)0x40)`
- `#define CAP_ITSource_TimerEqCompare ((uint32_t)0x80)`
- `#define CAP_ITSource_All ((uint32_t)0xFF)`
- `#define IS_CAP_IT_SOURCE(IT_SOURCE) (((IT_SOURCE) != (uint32_t)0x0000) && (((IT_SOURCE) & (uint32_t)0xFFFFF00) == ((uint32_t)0x00)))`
Макрос проверки источников прерываний на попадание в допустимый диапазон.
- `#define IS_CAP_IT_SOURCE_SINGLE(IT_SOURCE)`
Макрос проверки источника прерываний при работе с ними по отдельности (опрос флагов).

Перечисления

- enum `CAP_Capture_Polarity_TypeDef` { `CAP_Capture_Polarity_PosEdge`, `CAP_Capture_Polarity_NegEdge` }
Выбор фронта захвата.
- enum `CAP_Halt_TypeDef` { `CAP_Halt_Stop`, `CAP_Halt_StopOnZero`, `CAP_Halt_Free` }
Выбор режима остановки таймера при отладке.
- enum `CAP_SyncOut_TypeDef` { `CAP_SyncOut_Bypass`, `CAP_SyncOut_TimerEqPeriod`, `CAP_SyncOut_Disable` }
Выбор источника выходного сигнала синхронизации.
- enum `CAP_Capture_Mode_TypeDef` { `CAP_Capture_Mode_Cycle`, `CAP_Capture_Mode_Single` }
Выбор режима работы захвата.
- enum `CAP_PWM_Polarity_TypeDef` { `CAP_PWM_Polarity_Pos`, `CAP_PWM_Polarity_Neg` }
Выбор активного уровня в режиме ШИМ.
- enum `CAP_Mode_TypeDef` { `CAP_Mode_Capture`, `CAP_Mode_PWM` }
Выбор режима работы блока захвата.

Функции

- void `CAP_DeInit` (`NT_CAP_TypeDef *CAPx`)
Устанавливает все регистры блока захвата значениями по умолчанию.
- void `CAP_Init` (`NT_CAP_TypeDef *CAPx`, `CAP_Init_TypeDef *CAP_InitStruct`)
Инициализирует `CAPx` согласно параметрам структуры `CAP_InitStruct`.
- void `CAP_StructInit` (`CAP_Init_TypeDef *CAP_InitStruct`)
Заполнение каждого члена структуры `CAP_InitStruct` значениями по умолчанию.
- void `CAP_TimerCmd` (`NT_CAP_TypeDef *CAPx`, `FunctionalState State`)
Разрешение работы таймера, выбранного блока захвата.
- void `CAP_SetTimer` (`NT_CAP_TypeDef *CAPx`, `uint32_t TimerVal`)
Установка текущего значения счетчика напрямую.
- void `CAP_SetShadowTimer` (`NT_CAP_TypeDef *CAPx`, `uint32_t TimerVal`)
Установка теневого значения таймера для отложенной записи.
- `uint32_t` `CAP_GetTimer` (`NT_CAP_TypeDef *CAPx`)
Получение текущего значения таймера.
- `uint32_t` `CAP_GetShadowTimer` (`NT_CAP_TypeDef *CAPx`)
Получение отложенного значения таймера.
- void `CAP_SyncCmd` (`NT_CAP_TypeDef *CAPx`, `FunctionalState State`)
Разрешение синхронизации.
- void `CAP_SwSync` (`NT_CAP_TypeDef *CAPx`)
Проведение программной синхронизации.
- void `CAP_PWM_Init` (`NT_CAP_TypeDef *CAPx`, `CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct`)
Инициализирует режим ШИМ блока `CAPx` согласно параметрам структуры `CAP_PWM_InitStruct`.
- void `CAP_PWM_StructInit` (`CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct`)
Заполнение каждого члена структуры `CAP_PWM_InitStruct` значениями по умолчанию.
- void `CAP_PWM_SetPeriod` (`NT_CAP_TypeDef *CAPx`, `uint32_t PeriodVal`)
Установка значения периода ШИМ.
- void `CAP_PWM_SetCompare` (`NT_CAP_TypeDef *CAPx`, `uint32_t CompareVal`)
Установка значения сравнения ШИМ.
- void `CAP_PWM_SetShadowPeriod` (`NT_CAP_TypeDef *CAPx`, `uint32_t PeriodVal`)

- Установка значения периода ШИМ для отложенной записи.

 - void [CAP_PWM_SetShadowCompare](#) (NT_CAP_TypeDef *CAPx, uint32_t CompareVal)
- Установка значения сравнения ШИМ для отложенной записи.

 - uint32_t [CAP_PWM_GetPeriod](#) (NT_CAP_TypeDef *CAPx)
- Получение текущего периода ШИМ.

 - uint32_t [CAP_PWM_GetCompare](#) (NT_CAP_TypeDef *CAPx)
- Получение текущего значения сравнения ШИМ.

 - uint32_t [CAP_PWM_GetShadowPeriod](#) (NT_CAP_TypeDef *CAPx)
- Получение отложенного значения периода ШИМ.

 - uint32_t [CAP_PWM_GetShadowCompare](#) (NT_CAP_TypeDef *CAPx)
- Получение отложенного значения сравнения ШИМ.

 - void [CAP_Capture_Init](#) (NT_CAP_TypeDef *CAPx, [CAP_Capture_Init_TypeDef](#) *CAP_Capture_InitStruct)

Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.
- void [CAP_Capture_StructInit](#) ([CAP_Capture_Init_TypeDef](#) *CAP_Capture_InitStruct)

Заполнение каждого члена структуры CAP_Capture_InitStruct значениями по умолчанию.
- void [CAP_Capture_Cmd](#) (NT_CAP_TypeDef *CAPx, [FunctionalState](#) State)

Разрешение захвата для выбранного блока захвата.
- void [CAP_Capture_SetCap0](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)

Установка значения регистра захвата 0.
- void [CAP_Capture_SetCap1](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)

Установка значения регистра захвата 1.
- void [CAP_Capture_SetCap2](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)

Установка значения регистра захвата 2.
- void [CAP_Capture_SetCap3](#) (NT_CAP_TypeDef *CAPx, uint32_t Value)

Установка значения регистра захвата 3.
- uint32_t [CAP_Capture_GetCap0](#) (NT_CAP_TypeDef *CAPx)

Получение текущего значения из регистра захвата 0.
- uint32_t [CAP_Capture_GetCap1](#) (NT_CAP_TypeDef *CAPx)

Получение текущего значения из регистра захвата 1.
- uint32_t [CAP_Capture_GetCap2](#) (NT_CAP_TypeDef *CAPx)

Получение текущего значения из регистра захвата 2.
- uint32_t [CAP_Capture_GetCap3](#) (NT_CAP_TypeDef *CAPx)

Получение текущего значения из регистра захвата 3.
- void [CAP_ITCmd](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, [FunctionalState](#) State)

Разрешение работы прерывания выбранного блока захвата.
- void [CAP_ITForceCmd](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)

Принудительный вызов прерывания выбранного блока захвата.
- [FlagStatus](#) [CAP_ITStatus](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)

Чтение статуса флага источника прерывания выбранного блока захвата.
- void [CAP_ITStatusClear](#) (NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)

Сброс флагов источников прерываний выбранного блока захвата.
- [FlagStatus](#) [CAP_ITPendStatus](#) (NT_CAP_TypeDef *CAPx)

Чтение статуса прерывания выбранного блока захвата.
- void [CAP_ITPendClear](#) (NT_CAP_TypeDef *CAPx)

Сброс флага прерывания выбранного блока захвата.

8.7.1 Подробное описание

Файл содержит все прототипы функций для блоков захвата

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

18.01.2016

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2016 ОАО "НИИЭТ"

8.7.2 Макросы

8.7.2.1 `#define CAP_ITSource_All ((uint32_t)0xFF)`

Все источники выбраны.

См. определение в файле `niietcm4_cap.h` строка 252

8.7.2.2 `#define CAP_ITSource_CapEvent0 ((uint32_t)0x02)`

Событие захвата 0.

См. определение в файле `niietcm4_cap.h` строка 245

8.7.2.3 `#define CAP_ITSource_CapEvent1 ((uint32_t)0x04)`

Событие захвата 1.

См. определение в файле `niietcm4_cap.h` строка 246

8.7.2.4 `#define CAP_ITSource_CapEvent2 ((uint32_t)0x08)`

Событие захвата 2.

См. определение в файле `niietcm4_cap.h` строка 247

8.7.2.5 `#define CAP_ITSource_CapEvent3 ((uint32_t)0x10)`

Событие захвата 3.

См. определение в файле niietcm4_cap.h строка 248

8.7.2.6 `#define CAP_ITSource_GeneralInt ((uint32_t)0x01)`

Общее прерывание.

См. определение в файле niietcm4_cap.h строка 244

8.7.2.7 `#define CAP_ITSource_TimerEqCompare ((uint32_t)0x80)`

Счетчик таймера равен значению сравнения (в режиме ШИМ).

См. определение в файле niietcm4_cap.h строка 251

8.7.2.8 `#define CAP_ITSource_TimerEqPeriod ((uint32_t)0x40)`

Счетчик таймера равен периоду (в режиме ШИМ).

См. определение в файле niietcm4_cap.h строка 250

8.7.2.9 `#define CAP_ITSource_TimerOvf ((uint32_t)0x20)`

Переполнение счетчика таймера.

См. определение в файле niietcm4_cap.h строка 249

8.7.2.10 `#define IS_CAP_CAPTURE_MODE(CAPTURE_MODE)`

Макроопределение:

```
((CAPTURE_MODE) == CAP_Capture_Mode_Single) || \
((CAPTURE_MODE) == \
CAP_Capture_Mode_Cycle))
```

Макрос проверки аргументов типа `CAP_Capture_Mode_TypeDef`.

См. определение в файле niietcm4_cap.h строка 121

Используется в `CAP_Capture_Init()`.

8.7.2.11 `#define IS_CAP_CAPTURE_POLARITY(CAPTURE_POLARITY)`

Макроопределение:

```
((CAPTURE_POLARITY) == CAP_Capture_Polarity_PosEdge) || \
((CAPTURE_POLARITY) == \
CAP_Capture_Polarity_NegEdge))
```

Макрос проверки аргументов типа `CAP_Capture_Polarity_TypeDef`.

См. определение в файле niietcm4_cap.h строка 66

Используется в `CAP_Capture_Init()`.

8.7.2.12 #define IS_CAP_HALT(HALT)

Макроопределение:

```
((HALT) == CAP_Halt_Stop) || \
  ((HALT) == CAP_Halt_StopOnZero) || \
  ((HALT) == CAP_Halt_Free))
```

Макрос проверки аргументов типа CAP_Halt_TypeDef.

См. определение в файле niietcm4_cap.h строка 84

Используется в CAP_Init().

8.7.2.13 #define IS_CAP_IT_SOURCE_SINGLE(IT_SOURCE)

Макроопределение:

```
((IT_SOURCE) == CAP_ITSource_GeneralInt) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent0) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent1) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent2) || \
  ((IT_SOURCE) == CAP_ITSource_CapEvent3) || \
  ((IT_SOURCE) == CAP_ITSource_TimerOvf) || \
  ((IT_SOURCE) == CAP_ITSource_TimerEqPeriod) || \
  ((IT_SOURCE) == CAP_ITSource_TimerEqCompare))
```

Макрос проверки источника прерываний при работе с ними по отдельности (опрос флагов).

См. определение в файле niietcm4_cap.h строка 265

Используется в CAP_ITStatus().

8.7.2.14 #define IS_CAP_MODE(MODE)

Макроопределение:

```
((MODE) == CAP_Mode_Capture) || \
  ((MODE) == CAP_Mode_PWM))
```

Макрос проверки аргументов типа CAP_Mode_TypeDef.

См. определение в файле niietcm4_cap.h строка 155

Используется в CAP_Init().

8.7.2.15 #define IS_CAP_PWM_POLARITY(PWM_POLARITY)

Макроопределение:

```
((PWM_POLARITY) == CAP_PWM_Polarity_Pos) || \
  ((PWM_POLARITY) == CAP_PWM_Polarity_Neg))
```

Макрос проверки аргументов типа CAP_PWM_Polarity_TypeDef.

См. определение в файле niietcm4_cap.h строка 138

Используется в CAP_PWM_Init().

8.7.2.16 `#define IS_CAP_SYNC_OUT(SYNC_OUT)`

Макроопределение:

```
((SYNC_OUT) == CAP_SyncOut_Bypass) || \
((SYNC_OUT) == CAP_SyncOut_TimerEqPeriod) || \
((SYNC_OUT) == CAP_SyncOut_Disable))
```

Макрос проверки аргументов типа `CAP_SyncOut_TypeDef`.

См. определение в файле niietcm4_cap.h строка 103

Используется в `CAP_Init()`.

8.7.3 Перечисления

8.7.3.1 `enum CAP_Capture_Mode_TypeDef`

Выбор режима работы захвата.

Элементы перечислений

`CAP_Capture_Mode_Cycle` Циклический захват.

`CAP_Capture_Mode_Single` Однократный захват.

См. определение в файле niietcm4_cap.h строка 111

8.7.3.2 `enum CAP_Capture_Polarity_TypeDef`

Выбор фронта захвата.

Элементы перечислений

`CAP_Capture_Polarity_PosEdge` Захват по переднему фронту.

`CAP_Capture_Polarity_NegEdge` Захват по заднему фронту.

См. определение в файле niietcm4_cap.h строка 56

8.7.3.3 `enum CAP_Halt_TypeDef`

Выбор режима остановки таймера при отладке.

Элементы перечислений

`CAP_Halt_Stop` Мгновенная остановка таймера при отладке.

`CAP_Halt_StopOnZero` Остановка таймера при переполнении или сбросе (событие достижения 0).

`CAP_Halt_Free` Нормальный режим.

См. определение в файле niietcm4_cap.h строка 73

8.7.3.4 `enum CAP_Mode_TypeDef`

Выбор режима работы блока захвата.

Элементы перечислений

CAP_Mode_Capture Режим захвата.

CAP_Mode_PWM Режим ШИМ.

См. определение в файле niietcm4_cap.h строка 145

8.7.3.5 enum CAP_PWM_Polarity_TypeDef

Выбор активного уровня в режиме ШИМ.

Элементы перечислений

CAP_PWM_Polarity_Pos Высокий уровень является активным.

CAP_PWM_Polarity_Neg Низкий уровень является активным.

См. определение в файле niietcm4_cap.h строка 128

8.7.3.6 enum CAP_SyncOut_TypeDef

Выбор источника выходного сигнала синхронизации.

Элементы перечислений

CAP_SyncOut_Bypass Пропуск синхросигнала со входа на выход.

CAP_SyncOut_TimerEqPeriod Передача события равенства таймера и значения периода в качестве выходного сигнала синхронизации.

CAP_SyncOut_Disable Выходной сигнал синхронизации запрещен.

См. определение в файле niietcm4_cap.h строка 92

8.7.4 Функции

8.7.4.1 void CAP_Capture_Cmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение захвата для выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 444

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

8.7.4.2 uint32_t CAP_Capture_GetCap0 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 0.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 519

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.3 uint32_t CAP_Capture_GetCap1 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 1.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 532

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.4 uint32_t CAP_Capture_GetCap2 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 2.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 545

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.5 uint32_t CAP_Capture_GetCap3 (NT_CAP_TypeDef * CAPx)

Получение текущего значения из регистра захвата 3.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Value	Значение.
-------	-----------

См. определение в файле niietcm4_cap.c строка 558

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.6 void CAP_Capture_Init (NT_CAP_TypeDef * CAPx, CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)

Инициализирует режим захвата блока CAPx согласно параметрам структуры CAP_Capture_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_↵ Capture_Init↵ Struct	Указатель на структуру типа CAP_Capture_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 386

Перекрестные ссылки CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0, CAP_↵
Capture_Init_TypeDef::CAP_Capture_PolarityEvent1, CAP_Capture_Init_TypeDef::CAP_↵
Capture_PolarityEvent2, CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3, CAP_↵
Capture_Init_TypeDef::CAP_Capture_Prescale, CAP_Capture_Init_TypeDef::CAP_Capture_↵
RstEvent0, CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1, CAP_Capture_Init_Type↵
Def::CAP_Capture_RstEvent2, CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3, CAP_↵
Capture_Init_TypeDef::CAP_Capture_StopVal, CAP_Capture_Init_TypeDef::CAP_CaptureMode,
IS_CAP_ALL_PERIPH, IS_CAP_CAPTURE_MODE, IS_CAP_CAPTURE_POLARITY, IS_↵
CAP_CAPTURE_PRESCALE, IS_CAP_CAPTURE_STOP_VAL и IS_FUNCTIONAL_STATE.

8.7.4.7 void CAP_Capture_SetCap0 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 0.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 464

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.8 void CAP_Capture_SetCap1 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 1.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 478

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.9 void CAP_Capture_SetCap2 (NT_CAP_TypeDef * CAPx, uint32_t Value)

Установка значения регистра захвата 2.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 492

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

8.7.4.10 `void CAP_Capture_SetCap3 (NT_CAP_TypeDef * CAPx, uint32_t Value)`

Установка значения регистра захвата 3.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
Value	Значение.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 506

Перекрестные ссылки `IS_CAP_ALL_PERIPH`.

8.7.4.11 `void CAP_Capture_StructInit (CAP_Capture_Init_TypeDef * CAP_Capture_InitStruct)`

Заполнение каждого члена структуры `CAP_Capture_InitStruct` значениями по умолчанию.

Аргументы

CAP_Capture_InitStruct	Указатель на структуру типа CAP_Capture_Init_TypeDef , которую необходимо проинициализировать.
------------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_cap.c` строка 421

Перекрестные ссылки `CAP_Capture_Mode_Single`, `CAP_Capture_Polarity_PosEdge`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_PolarityEvent3`, `CAP_Capture_Init_TypeDef::CAP_Capture_Prescale`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent0`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent1`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent2`, `CAP_Capture_Init_TypeDef::CAP_Capture_RstEvent3`, `CAP_Capture_Init_TypeDef::CAP_Capture_StopVal` и `CAP_Capture_Init_TypeDef::CAP_CaptureMode`.

8.7.4.12 `void CAP_DeInit (NT_CAP_TypeDef * CAPx)`

Устанавливает все регистры блока захвата значениями по умолчанию.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 68

Перекрестные ссылки IS_CAP_ALL_PERIPH, RCC_PeriphRst_CAP0, RCC_PeriphRst_CAP1, RCC_PeriphRst_CAP2, RCC_PeriphRst_CAP3, RCC_PeriphRst_CAP4, RCC_PeriphRst_CAP5 и RCC_PeriphRstCmd().

8.7.4.13 uint32_t CAP_GetShadowTimer (NT_CAP_TypeDef * CAPx)

Получение отложенного значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 218

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.14 uint32_t CAP_GetTimer (NT_CAP_TypeDef * CAPx)

Получение текущего значения таймера.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

TimerVal	Значение таймера.
----------	-------------------

См. определение в файле niietcm4_cap.c строка 205

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.15 void CAP_Init (NT_CAP_TypeDef * CAPx, CAP_Init_TypeDef * CAP_InitStruct)

Инициализирует CAPx согласно параметрам структуры CAP_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef, которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 111

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Init_TypeDef::CAP_Mode, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut, IS_CAP_ALL_PERIPH, IS_CAP_HALT, IS_CAP_MODE и IS_CAP_SYNC_OUT.

8.7.4.16 void CAP_ITCmd (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource, FunctionalState State)

Разрешение работы прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT← Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 575

Перекрестные ссылки IS_CAP_ALL_PERIPH, IS_CAP_IT_SOURCE и IS_FUNCTIONAL_STATE.

8.7.4.17 void CAP_ITForceCmd (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Принудительный вызов прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT← Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 599

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

8.7.4.18 void CAP_ITPendClear (NT_CAP_TypeDef * CAPx)

Сброс флага прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 680

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.19 FlagStatus CAP_ITPendStatus (NT_CAP_TypeDef * CAPx)

Чтение статуса прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 656

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.20 FlagStatus CAP_ITStatus (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Чтение статуса флага источника прерывания выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_cap.c строка 615

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE_SINGLE.

8.7.4.21 void CAP_ITStatusClear (NT_CAP_TypeDef * CAPx, uint32_t CAP_ITSource)

Сброс флагов источников прерываний выбранного блока захвата.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из Маски источников прерываний .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 642

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_CAP_IT_SOURCE.

8.7.4.22 uint32_t CAP_PWM_GetCompare (NT_CAP_TypeDef * CAPx)

Получение текущего значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле niietcm4_cap.c строка 345

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.23 `uint32_t CAP_PWM_GetPeriod (NT_CAP_TypeDef * CAPx)`

Получение текущего периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле niietcm4_cap.c строка 332

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.24 uint32_t CAP_PWM_GetShadowCompare (NT_CAP_TypeDef * CAPx)

Получение отложенного значения сравнения ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

CompareVal	Значение сравнения.
------------	---------------------

См. определение в файле niietcm4_cap.c строка 371

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.25 uint32_t CAP_PWM_GetShadowPeriod (NT_CAP_TypeDef * CAPx)

Получение отложенного значения периода ШИМ.

Аргументы

CAPx	Выбор CAP, где x лежит в диапазоне 0-5.
------	---

Возвращаемые значения

PeriodVal	Значение периода.
-----------	-------------------

См. определение в файле niietcm4_cap.c строка 358

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.26 void CAP_PWM_Init (NT_CAP_TypeDef * CAPx, CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)

Инициализирует режим ШИМ блока CAPx согласно параметрам структуры CAP_PWM_InitStruct.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
CAP_PWM_InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef, которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_cap.c строка 246

Перекрестные ссылки CAP_PWM_Init_TypeDef::CAP_PWM_Compare, CAP_PWM_Init_TypeDef::CAP_PWM_Period, CAP_PWM_Init_TypeDef::CAP_PWM_Polarity, CAP_PWM_SetCompare(), CAP_PWM_SetPeriod(), IS_CAP_ALL_PERIPH и IS_CAP_PWM_POLARITY.

8.7.4.27 void CAP_PWM_SetCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 291

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

8.7.4.28 void CAP_PWM_SetPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 277

Перекрестные ссылки IS_CAP_ALL_PERIPH.

Используется в CAP_PWM_Init().

8.7.4.29 void CAP_PWM_SetShadowCompare (NT_CAP_TypeDef * CAPx, uint32_t CompareVal)

Установка значения сравнения ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
CompareVal	Значение сравнения.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 319

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.30 void CAP_PWM_SetShadowPeriod (NT_CAP_TypeDef * CAPx, uint32_t PeriodVal)

Установка значения периода ШИМ для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
PeriodVal	Значение периода.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 305

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.31 void CAP_PWM_StructInit (CAP_PWM_Init_TypeDef * CAP_PWM_InitStruct)

Заполнение каждого члена структуры CAP_PWM_InitStruct значениями по умолчанию.

Аргументы

CAP_PWM_↔ _InitStruct	Указатель на структуру типа CAP_PWM_Init_TypeDef, которую необходимо проинициализировать.
--------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 263

Перекрестные ссылки CAP_PWM_Init_TypeDef::CAP_PWM_Compare, CAP_PWM_Init_↔
TypeDef::CAP_PWM_Period, CAP_PWM_Init_TypeDef::CAP_PWM_Polarity и CAP_PWM_↔
Polarity_Pos.

8.7.4.32 void CAP_SetShadowTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)

Установка теневого значения таймера для отложенной записи.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 192

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.33 void CAP_SetTimer (NT_CAP_TypeDef * CAPx, uint32_t TimerVal)

Установка текущего значения счетчика напрямую.

Аргументы

CAPx	Выбор таймера, где x лежит в диапазоне 0-5.
TimerVal	Значение таймера.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 178

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.34 void CAP_StructInit (CAP_Init_TypeDef * CAP_InitStruct)

Заполнение каждого члена структуры CAP_InitStruct значениями по умолчанию.

Аргументы

CAP_InitStruct	Указатель на структуру типа CAP_Init_TypeDef, которую необходимо проинициализировать.
----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 147

Перекрестные ссылки CAP_Init_TypeDef::CAP_Halt, CAP_Halt_Stop, CAP_Init_TypeDef::CAP_Mode, CAP_Mode_Capture, CAP_Init_TypeDef::CAP_SyncCmd, CAP_Init_TypeDef::CAP_SyncOut и CAP_SyncOut_Bypass.

8.7.4.35 void CAP_SwSync (NT_CAP_TypeDef * CAPx)

Проведение программной синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 231

Перекрестные ссылки IS_CAP_ALL_PERIPH.

8.7.4.36 void CAP_SyncCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение синхронизации.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5
State	Выбор состояния. Параметр принимает любое значение из FunctionalState.

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 132

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

8.7.4.37 void CAP_TimerCmd (NT_CAP_TypeDef * CAPx, FunctionalState State)

Разрешение работы таймера, выбранного блока захвата.

Аргументы

CAPx	Выбор модуля CAP, где x лежит в диапазоне 0-5.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_cap.c строка 163

Перекрестные ссылки IS_CAP_ALL_PERIPH и IS_FUNCTIONAL_STATE.

8.8 Файл niietcm4_conf.h

Файл конфигурации драйвера.

```
#include "niietcm4_gpio.h"
#include "niietcm4_rcc.h"
#include "niietcm4_dma.h"
#include "niietcm4_uart.h"
#include "niietcm4_timer.h"
#include "niietcm4_rtc.h"
#include "niietcm4_bootflash.h"
#include "niietcm4_userflash.h"
#include "niietcm4_extmem.h"
#include "niietcm4_adc.h"
#include "niietcm4_watchdog.h"
#include "niietcm4_cap.h"
```

Макросы

- `#define assert_param(expr) ((void)0)`

8.8.1 Подробное описание

Файл конфигурации драйвера.

Основные функции:

- Исключение исходного кода для неиспользуемой периферии.
- Включение assert'ов.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.9 Файл niietcm4_dma.c

Файл содержит реализацию всех функции для работы с DMA.

```
#include "niietcm4_dma.h"
```

Функции

- void [DMA_ChannelDeInit](#) (DMA_Channel_TypeDef *DMA_Channel)
Деинициализация канала DMA.
- void [DMA_ChannelInit](#) (DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)
Инициализация канала DMA.
- void [DMA_ChannelStructInit](#) (DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)
Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.
- void [DMA_DeInit](#) ()
Деинициализация контроллера DMA.
- void [DMA_Init](#) (DMA_Init_TypeDef *DMA_InitStruct)
Инициализация контроллера DMA.
- void [DMA_StructInit](#) (DMA_Init_TypeDef *DMA_InitStruct)
Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.
- void [DMA_BasePtrConfig](#) (uint32_t BasePtr)
Установка базового адреса управляющих каналов.
- void [DMA_ProtectionConfig](#) (DMA_Protect_TypeDef *DMA_Protection)
Управление защитой шины при обращении DMA к управляющим данным.
- void [DMA_MasterEnableCmd](#) (FunctionalState State)
Разрешения работы контроллера DMA.
- void [DMA_SWRequestCmd](#) (uint32_t DMA_Channel)
Программный запрос на осуществление передач DMA по выбранным каналам.
- void [DMA_UseBurstCmd](#) (uint32_t DMA_Channel, FunctionalState State)
Установка пакетного обмена каналов DMA.
- void [DMA_ReqMaskCmd](#) (uint32_t DMA_Channel, FunctionalState State)

- Маскирование каналов DMA.
- void [DMA_ChannelEnableCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
- Активация каналов DMA.
- void [DMA_PrmAltCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
- Установка первичной/альтернативной управляющей структуры каналов DMA.
- void [DMA_HighPriorityCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
- Установка высокого приоритета каналов DMA.
- [DMA_State_TypeDef](#) [DMA_StateStatus](#) ()
- Доступ к текущему конечному автомату контроллера DMA.
- [FunctionalState](#) [DMA_MasterEnableStatus](#) ()
- Состояние контроллера DMA.
- [FunctionalState](#) [DMA_WaitOnReqStatus](#) (uint32_t DMA_Channel)
- Показывает поддерживает ли канал одиночные SREQ запросы.
- [OperationStatus](#) [DMA_ErrorStatus](#) ()
- Показывает наличие ошибки на шине.
- void [DMA_ClearErrorStatus](#) ()
- Сброс флага ошибки на шине.

8.9.1 Подробное описание

Файл содержит реализацию всех функции для работы с DMA.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

10.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.9.2 Функции

8.9.2.1 void DMA_BasePtrConfig (uint32_t BasePtr)

Установка базового адреса управляющих каналов.

Аргументы

BasePtr	Значение базового адреса.
---------	---------------------------

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 231

8.9.2.2 `void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)`

Деинициализация канала DMA.

Аргументы

DMA_Channel	Указатель на структуру типа DMA_Channel_TypeDef , которая содержит конфигурационную информацию канала.
-------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 87

Перекрестные ссылки `DMA_Channel_TypeDef::CHANNEL_CFG`, `DMA_Channel_TypeDef::DST↔_DATA_END` и `DMA_Channel_TypeDef::SRC_DATA_END`.

8.9.2.3 `void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)`

Активация каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 373

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

8.9.2.4 `void DMA_ChannelInit (DMA_Channel_TypeDef * DMA_Channel, DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)`

Инициализация канала DMA.

Аргументы

DMA_Channel	Непосредственно сама структура канала.
DMA↔ChannelInit↔Struct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которая содержит конфигурационную информацию канала.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 102

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Channel_TypeDef::CHANNEL_CFG_bit, CHANNEL_CFG_bits::CYCLE_CTRL, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal, DMA_Channel_TypeDef::DST_DATA_END, CHANNEL_CFG_bits::DST_INC, CHANNEL_CFG_bits::DST_PROT_BUFFERABLE, CHANNEL_CFG_bits::DST_PROT_CACHEABLE, CHANNEL_CFG_bits::DST_PROT_PRIVILEGED, CHANNEL_CFG_bits::DST_SIZE, IS_DMA_ARBITRATION_RATE, IS_DMA_DATA_INC, IS_DMA_DATA_SIZE, IS_DMA_MODE, IS_DMA_TRANSFERS_TOTAL, IS_FUNCTIONAL_STATE, CHANNEL_CFG_bits::N_MINUS_1, CHANNEL_CFG_bits::NEXT_USEBURST, DMA_Protect_TypeDef::PRIVELGED, CHANNEL_CFG_bits::R_POWER, DMA_Channel_TypeDef::SRC_DATA_END, CHANNEL_CFG_bits::SRC_INC, CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE, CHANNEL_CFG_bits::SRC_PROT_CACHEABLE, CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED и CHANNEL_CFG_bits::SRC_SIZE.

8.9.2.5 void DMA_ChannelStructInit (DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

Аргументы

DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которую необходимо проинициализировать.
-----------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 146

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ArbitrationRate_1, DMA_DataInc_Disable, DMA_DataSize_8, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_Mode_Disable, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal и DMA_Protect_TypeDef::PRIVELGED.

8.9.2.6 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 524

8.9.2.7 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 174

8.9.2.8 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

Status	Одно из значений OperationStatus: <ul style="list-style-type: none"> • OK - ошибок не было; • ERROR - произошла ошибка.
--------	---

См. определение в файле `niietcm4_dma.c` строка 503

8.9.2.9 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.9.2.10 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

Прежде чем инициализировать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать базовый адрес управляющей структуры с помощью [DMA_BasePtrConfig](#).

Аргументы

DMA_Init↔ Struct	Указатель на структуру типа DMA_Init_TypeDef , которая содержит конфигурационную информацию.
---------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 195

Перекрестные ссылки [DMA_Init_TypeDef::DMA_Channel](#), [DMA_Init_TypeDef::DMA_ChannelEnable](#), [DMA_ChannelEnableCmd\(\)](#), [DMA_Init_TypeDef::DMA_HighPriority](#), [DMA_HighPriorityCmd\(\)](#), [DMA_Init_TypeDef::DMA_PrmAlt](#), [DMA_PrmAltCmd\(\)](#), [DMA_Init_TypeDef::DMA_Protection](#), [DMA_ProtectionConfig\(\)](#), [DMA_Init_TypeDef::DMA_ReqMask](#), [DMA_ReqMaskCmd\(\)](#), [DMA_Init_TypeDef::DMA_UseBurst](#) и [DMA_UseBurstCmd\(\)](#).

8.9.2.11 void DMA_MasterEnableCmd (FunctionalState State)

Разрешения работы контроллера DMA.

Внимание

Прежде чем включать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать контроллер DMA через функцию инициализации [DMA_Init](#) или вручную - [Конфигурация контроллера DMA](#).

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 287

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

8.9.2.12 FunctionalState DMA_MasterEnableStatus ()

Состояние контроллера DMA.

Возвращаемые значения

Status	Текущее состояние контроллера DMA.
--------	------------------------------------

См. определение в файле niietcm4_dma.c строка 455

8.9.2.13 void DMA_PrmAltCmd (uint32_t DMA_Channel, FunctionalState State)

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 397

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

8.9.2.14 `void DMA_ProtectionConfig (DMA_Protect_TypeDef * DMA_Protection)`

Управление защитой шины при обращении DMA к управляющим данным.

Аргументы

<code>DMA_↔ Protection</code>	Структура, содержащая конфигурацию защиты. Параметр принимает структуру типа DMA_Protect_TypeDef .
-----------------------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 243

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHE↔
ABLE`, `IS_FUNCTIONAL_STATE` и `DMA_Protect_TypeDef::PRIVELGED`.

Используется в `DMA_Init()`.

8.9.2.15 `void DMA_ReqMaskCmd (uint32_t DMA_Channel, FunctionalState State)`

Маскирование каналов DMA.

Внимание

По маскированным каналам игнорируются запросы на передачи.

Аргументы

<code>DMA_Channel</code>	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 349

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

8.9.2.16 `DMA_State_TypeDef DMA_StateStatus ()`

Доступ к текущему конечного автомата контроллера DMA.

Возвращаемые значения

<code>State</code>	Текущее состояние конечного автомата.
--------------------	---------------------------------------

См. определение в файле `niietcm4_dma.c` строка 441

8.9.2.17 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

Аргументы

DMA_Init_↵ Struct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
----------------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 212

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHE↵
ABLE`, `DMA_Init_TypeDef::DMA_Channel`, `DMA_Init_TypeDef::DMA_ChannelEnable`, `DMA_↵
Init_TypeDef::DMA_HighPriority`, `DMA_Init_TypeDef::DMA_PrmAlt`, `DMA_Init_TypeDef::DM↵
A_Protection`, `DMA_Init_TypeDef::DMA_ReqMask`, `DMA_Init_TypeDef::DMA_UseBurst` и `DM↵
A_Protect_TypeDef::PRIVELGED`.

8.9.2.18 `void DMA_SWRequestCmd (uint32_t DMA_Channel)`

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски кана- лов по номеру .
-------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 308

Перекрестные ссылки `IS_DMA_CHANNEL`.

8.9.2.19 `void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка пакетного обмена каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски кана- лов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_dma.c` строка 324

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

8.9.2.20 `FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)`

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

Status	Одно из значений FunctionalState: <ul style="list-style-type: none"> • ENABLE - поддерживаются SREQ (как и блочные BREQ); • DISABLE - поддерживаются только блочные запросы BREQ.
--------	---

См. определение в файле niietcm4_dma.c строка 478

Перекрестные ссылки IS_GET_DMA_CHANNEL.

8.10 Файл niietcm4_dma.h

Файл содержит все прототипы функций для DMA.

```
#include "niietcm4.h"
```

Структуры данных

- struct [_CHANNEL_CFG_bits](#)
Битовый доступ к регистру CHANNEL_CFG в [DMA_Channel_TypeDef](#).
- struct [DMA_Channel_TypeDef](#)
Тип, описывающий структуру канала DMA.
- struct [DMA_ConfigStruct_TypeDef](#)
Управляющая структура данных DMA.
- struct [DMA_ConfigData_TypeDef](#)
Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ.
- struct [DMA_Protect_TypeDef](#)
Защита шины при чтении из источника или записи в приемник через DMA.
- struct [DMA_ChannelInit_TypeDef](#)
Структура инициализации канала DMA.
- struct [DMA_Init_TypeDef](#)
Структура инициализации контроллера DMA.

Макросы

- [#define CHANNEL_CFG_CYCLE_CTRL_Pos 0](#)
- [#define CHANNEL_CFG_NEXT_USEBURST_Pos 3](#)
- [#define CHANNEL_CFG_N_MINUS_1_Pos 4](#)
- [#define CHANNEL_CFG_R_POWER_Pos 14](#)
- [#define CHANNEL_CFG_SRC_PROT_CTRL_Pos 18](#)
- [#define CHANNEL_CFG_DST_PROT_CTRL_Pos 21](#)
- [#define CHANNEL_CFG_SRC_SIZE_Pos 24](#)
- [#define CHANNEL_CFG_SRC_INC_Pos 26](#)
- [#define CHANNEL_CFG_DST_SIZE_Pos 28](#)
- [#define CHANNEL_CFG_DST_INC_Pos 30](#)
- [#define CHANNEL_CFG_CYCLE_CTRL_Msk \(\(uint32_t\)0x00000007\)](#)
- [#define CHANNEL_CFG_NEXT_USEBURST_Msk \(\(uint32_t\)0x00000008\)](#)
- [#define CHANNEL_CFG_N_MINUS_1_Msk \(\(uint32_t\)0x00003FF0\)](#)
- [#define CHANNEL_CFG_R_POWER_Msk \(\(uint32_t\)0x0003C000\)](#)
- [#define CHANNEL_CFG_SRC_PROT_CTRL_Msk \(\(uint32_t\)0x001C0000\)](#)
- [#define CHANNEL_CFG_DST_PROT_CTRL_Msk \(\(uint32_t\)0x00E00000\)](#)

```

• #define CHANNEL_CFG_SRC_SIZE_Msk ((uint32_t)0x03000000)
• #define CHANNEL_CFG_SRC_INC_Msk ((uint32_t)0x0C000000)
• #define CHANNEL_CFG_DST_SIZE_Msk ((uint32_t)0x30000000)
• #define CHANNEL_CFG_DST_INC_Msk ((uint32_t)0xC0000000)
• #define DMA_Channel_All ((uint32_t)0x00FFFFFF)
• #define DMA_Channel_UART0_TX ((uint32_t)0x00000001)
• #define DMA_Channel_UART1_TX ((uint32_t)0x00000002)
• #define DMA_Channel_UART2_TX ((uint32_t)0x00000004)
• #define DMA_Channel_UART3_TX ((uint32_t)0x00000008)
• #define DMA_Channel_UART0_RX ((uint32_t)0x00000010)
• #define DMA_Channel_UART1_RX ((uint32_t)0x00000020)
• #define DMA_Channel_UART2_RX ((uint32_t)0x00000040)
• #define DMA_Channel_UART3_RX ((uint32_t)0x00000080)
• #define DMA_Channel_ADCSEQ0 ((uint32_t)0x00000100)
• #define DMA_Channel_ADCSEQ1 ((uint32_t)0x00000200)
• #define DMA_Channel_ADCSEQ2 ((uint32_t)0x00000400)
• #define DMA_Channel_ADCSEQ3 ((uint32_t)0x00000800)
• #define DMA_Channel_ADCSEQ4 ((uint32_t)0x00001000)
• #define DMA_Channel_ADCSEQ5 ((uint32_t)0x00002000)
• #define DMA_Channel_ADCSEQ6 ((uint32_t)0x00004000)
• #define DMA_Channel_ADCSEQ7 ((uint32_t)0x00008000)
• #define DMA_Channel_SPI0_TX ((uint32_t)0x00010000)
• #define DMA_Channel_SPI1_TX ((uint32_t)0x00020000)
• #define DMA_Channel_SPI2_TX ((uint32_t)0x00040000)
• #define DMA_Channel_SPI3_TX ((uint32_t)0x00080000)
• #define DMA_Channel_SPI0_RX ((uint32_t)0x00100000)
• #define DMA_Channel_SPI1_RX ((uint32_t)0x00200000)
• #define DMA_Channel_SPI2_RX ((uint32_t)0x00400000)
• #define DMA_Channel_SPI3_RX ((uint32_t)0x00800000)
• #define DMA_Channel_0 ((uint32_t)0x00000001)
• #define DMA_Channel_1 ((uint32_t)0x00000002)
• #define DMA_Channel_2 ((uint32_t)0x00000004)
• #define DMA_Channel_3 ((uint32_t)0x00000008)
• #define DMA_Channel_4 ((uint32_t)0x00000010)
• #define DMA_Channel_5 ((uint32_t)0x00000020)
• #define DMA_Channel_6 ((uint32_t)0x00000040)
• #define DMA_Channel_7 ((uint32_t)0x00000080)
• #define DMA_Channel_8 ((uint32_t)0x00000100)
• #define DMA_Channel_9 ((uint32_t)0x00000200)
• #define DMA_Channel_10 ((uint32_t)0x00000400)
• #define DMA_Channel_11 ((uint32_t)0x00000800)
• #define DMA_Channel_12 ((uint32_t)0x00001000)
• #define DMA_Channel_13 ((uint32_t)0x00002000)
• #define DMA_Channel_14 ((uint32_t)0x00004000)
• #define DMA_Channel_15 ((uint32_t)0x00008000)
• #define DMA_Channel_16 ((uint32_t)0x00010000)
• #define DMA_Channel_17 ((uint32_t)0x00020000)
• #define DMA_Channel_18 ((uint32_t)0x00040000)
• #define DMA_Channel_19 ((uint32_t)0x00080000)
• #define DMA_Channel_20 ((uint32_t)0x00100000)
• #define DMA_Channel_21 ((uint32_t)0x00200000)
• #define DMA_Channel_22 ((uint32_t)0x00400000)
• #define DMA_Channel_23 ((uint32_t)0x00800000)
• #define IS_DMA_CHANNEL(CHANNEL) (((CHANNEL) != (uint32_t)0x000000) && (((CHANNEL) & (uint32_t)0xFF000000) == ((uint32_t)0x0000)))

```


- Макрос проверки маски каналов на попадание в допустимый диапазон.
- `#define IS_GET_DMA_CHANNEL(CHANNEL)`
Макрос проверки маски канала при работе с каналами по отдельности.
- `#define IS_DMA_MODE(MODE)`
Макрос проверки аргументов типа `DMA_Mode_TypeDef`.
- `#define IS_DMA_ARBITRATION_RATE(ARBITRATION_RATE)`
Макрос проверки аргументов типа `DMA_ArbitrationRate_TypeDef`.
- `#define IS_DMA_DATA_SIZE(DATA_SIZE)`
Макрос проверки аргументов типа `DMA_DataSize_TypeDef`.
- `#define IS_DMA_DATA_INC(DATA_INC)`
Макрос проверки аргументов типа `DMA_DataSize_TypeDef`.
- `#define IS_DMA_TRANSFERS_TOTAL(TRANSFERS_TOTAL) (((TRANSFERS_TOTAL) <= ((uint32_t)1024)) && ((TRANSFERS_TOTAL) >= ((uint32_t)1)))`
Макрос проверки соответствия величины `DMA_TransfersTotal` из `DMA_ChannelInit_TypeDef` разрешенному диапазону.
- `#define IS_DMA_STATE(STATE)`
Макрос проверки аргументов типа `DMA_State_TypeDef`.

Перечисления

- `enum DMA_Mode_TypeDef {`
`DMA_Mode_Disable, DMA_Mode_Basic, DMA_Mode_AutoReq, DMA_Mode_PingPong,`
`DMA_Mode_PrmMemScatGath, DMA_Mode_AltMemScatGath, DMA_Mode_PrmPeriphScatGath, DMA_Mode_AltPeriphScatGath }`
 Выбор режима работы DMA.
- `enum DMA_ArbitrationRate_TypeDef {`
`DMA_ArbitrationRate_1, DMA_ArbitrationRate_2, DMA_ArbitrationRate_4, DMA_ArbitrationRate_8,`
`DMA_ArbitrationRate_16, DMA_ArbitrationRate_32, DMA_ArbitrationRate_64, DMA_ArbitrationRate_128,`
`DMA_ArbitrationRate_256, DMA_ArbitrationRate_512, DMA_ArbitrationRate_1024 }`
 Выбор количества передач до выполнения переарбитрации.
- `enum DMA_DataSize_TypeDef { DMA_DataSize_8, DMA_DataSize_16, DMA_DataSize_32 }`
 Разрядность данных источника или приемника
- `enum DMA_DataInc_TypeDef { DMA_DataInc_8, DMA_DataInc_16, DMA_DataInc_32, DMA_DataInc_Disable }`
 Шаг инкремента адреса источника при чтении или приемника при записи
- `enum DMA_State_TypeDef {`
`DMA_State_Free, DMA_State_ReadConfigData, DMA_State_ReadSrcDataEndPtr, DMA_State_ReadDstDataEndPtr,`
`DMA_State_ReadSrcData, DMA_State_WriteDstData, DMA_State_WaitReq, DMA_State_WriteConfigData,`
`DMA_State_Pause, DMA_State_Done, DMA_State_PeriphScatGath }`
 Возможные состояния конечного автомата управления контроллером DMA.

Функции

- `void DMA_ChannelDeInit (DMA_Channel_TypeDef *DMA_Channel)`
Деинициализация канала DMA.
- `void DMA_ChannelInit (DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)`
Инициализация канала DMA.

- void [DMA_ChannelStructInit](#) ([DMA_ChannelInit_TypeDef](#) *DMA_ChannelInitStruct)
Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.
- void [DMA_DeInit](#) ()
Деинициализация контроллера DMA.
- void [DMA_Init](#) ([DMA_Init_TypeDef](#) *DMA_InitStruct)
Инициализация контроллера DMA.
- void [DMA_StructInit](#) ([DMA_Init_TypeDef](#) *DMA_InitStruct)
Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.
- void [DMA_BasePtrConfig](#) (uint32_t BasePtr)
Установка базового адреса управляющих каналов.
- void [DMA_ProtectionConfig](#) ([DMA_Protect_TypeDef](#) *DMA_Protection)
Управление защитой шины при обращении DMA к управляющим данным.
- void [DMA_MasterEnableCmd](#) ([FunctionalState](#) State)
Разрешения работы контроллера DMA.
- void [DMA_SWRequestCmd](#) (uint32_t DMA_Channel)
Программный запрос на осуществление передач DMA по выбранным каналам.
- void [DMA_UseBurstCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Установка пакетного обмена каналов DMA.
- void [DMA_ReqMaskCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Маскирование каналов DMA.
- void [DMA_ChannelEnableCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Активация каналов DMA.
- void [DMA_PrmAltCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Установка первичной/альтернативной управляющей структуры каналов DMA.
- void [DMA_HighPriorityCmd](#) (uint32_t DMA_Channel, [FunctionalState](#) State)
Установка высокого приоритета каналов DMA.
- [DMA_State_TypeDef](#) [DMA_StateStatus](#) ()
Доступ к текущему конечного автомата контроллера DMA.
- [FunctionalState](#) [DMA_MasterEnableStatus](#) ()
Состояние контроллера DMA.
- [FunctionalState](#) [DMA_WaitOnReqStatus](#) (uint32_t DMA_Channel)
Показывает поддерживает ли канал одиночные SREQ запросы.
- [OperationStatus](#) [DMA_ErrorStatus](#) ()
Показывает наличие ошибки на шине.
- void [DMA_ClearErrorStatus](#) ()
Сброс флага ошибки на шине.

8.10.1 Подробное описание

Файл содержит все прототипы функций для DMA.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

10.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.10.2 Макросы

8.10.2.1 `#define CHANNEL_CFG_CYCLE_CTRL_Msk ((uint32_t)0x00000007)`

Поле задания типа цикла DMA

См. определение в файле niietcm4_dma.h строка 68

8.10.2.2 `#define CHANNEL_CFG_CYCLE_CTRL_Pos 0`

Поле задания типа цикла DMA

См. определение в файле niietcm4_dma.h строка 57

8.10.2.3 `#define CHANNEL_CFG_DST_INC_Msk ((uint32_t)0xC0000000)`

Шаг инкремента адреса приемника

См. определение в файле niietcm4_dma.h строка 77

8.10.2.4 `#define CHANNEL_CFG_DST_INC_Pos 30`

Шаг инкремента адреса приемника

См. определение в файле niietcm4_dma.h строка 66

8.10.2.5 `#define CHANNEL_CFG_DST_PROT_CTRL_Msk ((uint32_t)0x00E00000)`

Защита шины АНВ-Lite при записи данных в приемник

См. определение в файле niietcm4_dma.h строка 73

8.10.2.6 `#define CHANNEL_CFG_DST_PROT_CTRL_Pos 21`

Защита шины АНВ-Lite при записи данных в приемник

См. определение в файле niietcm4_dma.h строка 62

8.10.2.7 `#define CHANNEL_CFG_DST_SIZE_Msk ((uint32_t)0x30000000)`

Разрядность данных приемника

См. определение в файле `niietcm4_dma.h` строка 76

8.10.2.8 `#define CHANNEL_CFG_DST_SIZE_Pos 28`

Разрядность данных приемника

См. определение в файле `niietcm4_dma.h` строка 65

8.10.2.9 `#define CHANNEL_CFG_N_MINUS_1_Msk ((uint32_t)0x00003FF0)`

Общее количество передач в цикле работы

См. определение в файле `niietcm4_dma.h` строка 70

8.10.2.10 `#define CHANNEL_CFG_N_MINUS_1_Pos 4`

Общее количество передач в цикле работы

См. определение в файле `niietcm4_dma.h` строка 59

8.10.2.11 `#define CHANNEL_CFG_NEXT_USEBURST_Msk ((uint32_t)0x00000008)`

Контролирует установку соответствующего каналу бита в регистре `NT_DMA->CHNL_USEBURST_SET`

См. определение в файле `niietcm4_dma.h` строка 69

8.10.2.12 `#define CHANNEL_CFG_NEXT_USEBURST_Pos 3`

Контролирует установку соответствующего каналу бита в регистре `NT_DMA->CHNL_USEBURST_SET`

См. определение в файле `niietcm4_dma.h` строка 58

8.10.2.13 `#define CHANNEL_CFG_R_POWER_Msk ((uint32_t)0x0003C000)`

Количество передач до выполнения переарбитрации

См. определение в файле `niietcm4_dma.h` строка 71

8.10.2.14 `#define CHANNEL_CFG_R_POWER_Pos 14`

Количество передач до выполнения переарбитрации

См. определение в файле `niietcm4_dma.h` строка 60

8.10.2.15 `#define CHANNEL_CFG_SRC_INC_Msk ((uint32_t)0x0C000000)`

Шаг инкремента адреса источника

См. определение в файле `niietcm4_dma.h` строка 75

8.10.2.16 `#define CHANNEL_CFG_SRC_INC_Pos 26`

Шаг инкремента адреса источника

См. определение в файле niietcm4_dma.h строка 64

8.10.2.17 `#define CHANNEL_CFG_SRC_PROT_CTRL_Msk ((uint32_t)0x001C0000)`

Защита шины АHB-Lite при чтении данных из источника

См. определение в файле niietcm4_dma.h строка 72

8.10.2.18 `#define CHANNEL_CFG_SRC_PROT_CTRL_Pos 18`

Защита шины АHB-Lite при чтении данных из источника

См. определение в файле niietcm4_dma.h строка 61

8.10.2.19 `#define CHANNEL_CFG_SRC_SIZE_Msk ((uint32_t)0x03000000)`

Разрядность данных источника

См. определение в файле niietcm4_dma.h строка 74

8.10.2.20 `#define CHANNEL_CFG_SRC_SIZE_Pos 24`

Разрядность данных источника

См. определение в файле niietcm4_dma.h строка 63

8.10.2.21 `#define DMA_Channel_0 ((uint32_t)0x00000001)`

Канал DMA 0

См. определение в файле niietcm4_dma.h строка 126

8.10.2.22 `#define DMA_Channel_1 ((uint32_t)0x00000002)`

Канал DMA 1

См. определение в файле niietcm4_dma.h строка 127

8.10.2.23 `#define DMA_Channel_10 ((uint32_t)0x00000400)`

Канал DMA 10

См. определение в файле niietcm4_dma.h строка 136

8.10.2.24 `#define DMA_Channel_11 ((uint32_t)0x00000800)`

Канал DMA 11

См. определение в файле niietcm4_dma.h строка 137

8.10.2.25 `#define DMA_Channel_12 ((uint32_t)0x00001000)`

Канал DMA 12

См. определение в файле `niietcm4_dma.h` строка 138

8.10.2.26 `#define DMA_Channel_13 ((uint32_t)0x00002000)`

Канал DMA 13

См. определение в файле `niietcm4_dma.h` строка 139

8.10.2.27 `#define DMA_Channel_14 ((uint32_t)0x00004000)`

Канал DMA 14

См. определение в файле `niietcm4_dma.h` строка 140

8.10.2.28 `#define DMA_Channel_15 ((uint32_t)0x00008000)`

Канал DMA 15

См. определение в файле `niietcm4_dma.h` строка 141

8.10.2.29 `#define DMA_Channel_16 ((uint32_t)0x00010000)`

Канал DMA 16

См. определение в файле `niietcm4_dma.h` строка 142

8.10.2.30 `#define DMA_Channel_17 ((uint32_t)0x00020000)`

Канал DMA 17

См. определение в файле `niietcm4_dma.h` строка 143

8.10.2.31 `#define DMA_Channel_18 ((uint32_t)0x00040000)`

Канал DMA 18

См. определение в файле `niietcm4_dma.h` строка 144

8.10.2.32 `#define DMA_Channel_19 ((uint32_t)0x00080000)`

Канал DMA 19

См. определение в файле `niietcm4_dma.h` строка 145

8.10.2.33 `#define DMA_Channel_2 ((uint32_t)0x00000004)`

Канал DMA 2

См. определение в файле `niietcm4_dma.h` строка 128

8.10.2.34 `#define DMA_Channel_20 ((uint32_t)0x00100000)`

Канал DMA 20

См. определение в файле `niietcm4_dma.h` строка 146

8.10.2.35 `#define DMA_Channel_21 ((uint32_t)0x00200000)`

Канал DMA 21

См. определение в файле niietcm4_dma.h строка 147

8.10.2.36 `#define DMA_Channel_22 ((uint32_t)0x00400000)`

Канал DMA 22

См. определение в файле niietcm4_dma.h строка 148

8.10.2.37 `#define DMA_Channel_23 ((uint32_t)0x00800000)`

Канал DMA 23

См. определение в файле niietcm4_dma.h строка 149

8.10.2.38 `#define DMA_Channel_3 ((uint32_t)0x00000008)`

Канал DMA 3

См. определение в файле niietcm4_dma.h строка 129

8.10.2.39 `#define DMA_Channel_4 ((uint32_t)0x00000010)`

Канал DMA 4

См. определение в файле niietcm4_dma.h строка 130

8.10.2.40 `#define DMA_Channel_5 ((uint32_t)0x00000020)`

Канал DMA 5

См. определение в файле niietcm4_dma.h строка 131

8.10.2.41 `#define DMA_Channel_6 ((uint32_t)0x00000040)`

Канал DMA 6

См. определение в файле niietcm4_dma.h строка 132

8.10.2.42 `#define DMA_Channel_7 ((uint32_t)0x00000080)`

Канал DMA 7

См. определение в файле niietcm4_dma.h строка 133

8.10.2.43 `#define DMA_Channel_8 ((uint32_t)0x00000100)`

Канал DMA 8

См. определение в файле niietcm4_dma.h строка 134

8.10.2.44 `#define DMA_Channel_9 ((uint32_t)0x00000200)`

Канал DMA 9

См. определение в файле niietcm4_dma.h строка 135

8.10.2.45 `#define DMA_Channel_ADCSEQ0 ((uint32_t)0x00000100)`

Канал DMA секвенсора 0 АЦП

См. определение в файле niietcm4_dma.h строка 101

8.10.2.46 `#define DMA_Channel_ADCSEQ1 ((uint32_t)0x00000200)`

Канал DMA секвенсора 1 АЦП

См. определение в файле niietcm4_dma.h строка 102

8.10.2.47 `#define DMA_Channel_ADCSEQ2 ((uint32_t)0x00000400)`

Канал DMA секвенсора 2 АЦП

См. определение в файле niietcm4_dma.h строка 103

8.10.2.48 `#define DMA_Channel_ADCSEQ3 ((uint32_t)0x00000800)`

Канал DMA секвенсора 3 АЦП

См. определение в файле niietcm4_dma.h строка 104

8.10.2.49 `#define DMA_Channel_ADCSEQ4 ((uint32_t)0x00001000)`

Канал DMA секвенсора 4 АЦП

См. определение в файле niietcm4_dma.h строка 105

8.10.2.50 `#define DMA_Channel_ADCSEQ5 ((uint32_t)0x00002000)`

Канал DMA секвенсора 5 АЦП

См. определение в файле niietcm4_dma.h строка 106

8.10.2.51 `#define DMA_Channel_ADCSEQ6 ((uint32_t)0x00004000)`

Канал DMA секвенсора 6 АЦП

См. определение в файле niietcm4_dma.h строка 107

8.10.2.52 `#define DMA_Channel_ADCSEQ7 ((uint32_t)0x00008000)`

Канал DMA секвенсора 7 АЦП

См. определение в файле niietcm4_dma.h строка 108

8.10.2.53 `#define DMA_Channel_All ((uint32_t)0x00FFFFFF)`

Все каналы DMA

См. определение в файле niietcm4_dma.h строка 87

8.10.2.54 `#define DMA_Channel_SPI0_RX ((uint32_t)0x00100000)`

Канал DMA по приему от SPI0

См. определение в файле niietcm4_dma.h строка 113

8.10.2.55 `#define DMA_Channel_SPI0_TX ((uint32_t)0x00010000)`

Канал DMA по передаче от SPI0

См. определение в файле niietcm4_dma.h строка 109

8.10.2.56 `#define DMA_Channel_SPI1_RX ((uint32_t)0x00200000)`

Канал DMA по приему от SPI1

См. определение в файле niietcm4_dma.h строка 114

8.10.2.57 `#define DMA_Channel_SPI1_TX ((uint32_t)0x00020000)`

Канал DMA по передаче от SPI1

См. определение в файле niietcm4_dma.h строка 110

8.10.2.58 `#define DMA_Channel_SPI2_RX ((uint32_t)0x00400000)`

Канал DMA по приему от SPI2

См. определение в файле niietcm4_dma.h строка 115

8.10.2.59 `#define DMA_Channel_SPI2_TX ((uint32_t)0x00040000)`

Канал DMA по передаче от SPI2

См. определение в файле niietcm4_dma.h строка 111

8.10.2.60 `#define DMA_Channel_SPI3_RX ((uint32_t)0x00800000)`

Канал DMA по приему от SPI3

См. определение в файле niietcm4_dma.h строка 116

8.10.2.61 `#define DMA_Channel_SPI3_TX ((uint32_t)0x00080000)`

Канал DMA по передаче от SPI3

См. определение в файле niietcm4_dma.h строка 112

8.10.2.62 `#define DMA_Channel_UART0_RX ((uint32_t)0x00000010)`

Канал DMA по приему от UART0

См. определение в файле niietcm4_dma.h строка 97

8.10.2.63 `#define DMA_Channel_UART0_TX ((uint32_t)0x00000001)`

Канал DMA по передаче от UART0

См. определение в файле niietcm4_dma.h строка 93

8.10.2.64 `#define DMA_Channel_UART1_RX ((uint32_t)0x00000020)`

Канал DMA по приему от UART1

См. определение в файле niietcm4_dma.h строка 98

8.10.2.65 `#define DMA_Channel_UART1_TX ((uint32_t)0x00000002)`

Канал DMA по передаче от UART1

См. определение в файле niietcm4_dma.h строка 94

8.10.2.66 `#define DMA_Channel_UART2_RX ((uint32_t)0x00000040)`

Канал DMA по приему от UART2

См. определение в файле niietcm4_dma.h строка 99

8.10.2.67 `#define DMA_Channel_UART2_TX ((uint32_t)0x00000004)`

Канал DMA по передаче от UART2

См. определение в файле niietcm4_dma.h строка 95

8.10.2.68 `#define DMA_Channel_UART3_RX ((uint32_t)0x00000080)`

Канал DMA по приему от UART3

См. определение в файле niietcm4_dma.h строка 100

8.10.2.69 `#define DMA_Channel_UART3_TX ((uint32_t)0x00000008)`

Канал DMA по передаче от UART3

См. определение в файле niietcm4_dma.h строка 96

8.10.2.70 `#define IS_DMA_ARBITRATION_RATE(ARBITRATION_RATE)`

Макроопределение:

```
((ARBITRATION_RATE) == DMA_ArbitrationRate_1) || \
DMA_ArbitrationRate_2) || \
DMA_ArbitrationRate_4) || \
DMA_ArbitrationRate_8) || \
DMA_ArbitrationRate_16) || \
DMA_ArbitrationRate_32) || \
DMA_ArbitrationRate_64) || \
DMA_ArbitrationRate_128) || \
DMA_ArbitrationRate_256) || \
DMA_ArbitrationRate_512) || \
DMA_ArbitrationRate_1024))
```

Макрос проверки аргументов типа [DMA_ArbitrationRate_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 316

Используется в DMA_ChannelInit().

8.10.2.71 `#define IS_DMA_DATA_INC(DATA_INC)`

Макроопределение:

```
((DATA_INC) == DMA_DataInc_8) || \
((DATA_INC) == DMA_DataInc_16) || \
((DATA_INC) == DMA_DataInc_32) || \
((DATA_INC) == DMA_DataInc_Disable))
```

Макрос проверки аргументов типа [DMA_DataSize_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 374

Используется в DMA_ChannelInit().

8.10.2.72 `#define IS_DMA_DATA_SIZE(DATA_SIZE)`

Макроопределение:

```
((DATA_SIZE) == DMA_DataSize_8) || \
((DATA_SIZE) == DMA_DataSize_16) || \
((DATA_SIZE) == DMA_DataSize_32))
```

Макрос проверки аргументов типа [DMA_DataSize_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 354

Используется в DMA_ChannelInit().

8.10.2.73 `#define IS_DMA_MODE(MODE)`

Макроопределение:

```
((MODE) == DMA_Mode_Disable) || \
((MODE) == DMA_Mode_Basic) || \
((MODE) == DMA_Mode_AutoReq) || \
((MODE) == DMA_Mode_PingPong) || \
((MODE) == DMA_Mode_PrmMemScatGath) || \
((MODE) == DMA_Mode_AltMemScatGath) || \
((MODE) == DMA_Mode_PrmPeriphScatGath) || \
((MODE) == DMA_Mode_AltPeriphScatGath))
```

Макрос проверки аргументов типа [DMA_Mode_TypeDef](#).

См. определение в файле niietcm4_dma.h строка 284

Используется в DMA_ChannelInit().

8.10.2.74 `#define IS_DMA_STATE(STATE)`

Макроопределение:

```
((STATE) == DMA_State_Free) || \
((STATE) == DMA_State_ReadConfigData) || \
((STATE) == DMA_State_ReadSrcDataEndPtr) || \
((STATE) == DMA_State_ReadDstDataEndPtr) || \
((STATE) == DMA_State_ReadSrcData) || \
((STATE) == DMA_State_WriteDstData) || \
((STATE) == DMA_State_WaitReq) || \
((STATE) == DMA_State_Pause) || \
((STATE) == DMA_State_Done) || \
((STATE) == DMA_StatePeriphScatGath))
```

Макрос проверки аргументов типа `DMA_State_TypeDef`.

См. определение в файле `niietcm4_dma.h` строка 459

8.10.2.75 `#define IS_GET_DMA_CHANNEL(CHANNEL)`

Макроопределение:

```
((((CHANNEL) == (DMA_Channel_0 || DMA_Channel_UART0_TX)) || \
  ((CHANNEL) == (DMA_Channel_1 || \
    DMA_Channel_UART1_TX)) || \
  ((CHANNEL) == (DMA_Channel_2 || \
    DMA_Channel_UART2_TX)) || \
  ((CHANNEL) == (DMA_Channel_3 || \
    DMA_Channel_UART3_TX)) || \
  ((CHANNEL) == (DMA_Channel_4 || \
    DMA_Channel_UART0_RX)) || \
  ((CHANNEL) == (DMA_Channel_5 || \
    DMA_Channel_UART1_RX)) || \
  ((CHANNEL) == (DMA_Channel_6 || \
    DMA_Channel_UART2_RX)) || \
  ((CHANNEL) == (DMA_Channel_7 || \
    DMA_Channel_UART3_RX)) || \
  ((CHANNEL) == (DMA_Channel_8 || \
    DMA_Channel_ADCSEQ0)) || \
  ((CHANNEL) == (DMA_Channel_9 || \
    DMA_Channel_ADCSEQ1)) || \
  ((CHANNEL) == (DMA_Channel_10 || \
    DMA_Channel_ADCSEQ2)) || \
  ((CHANNEL) == (DMA_Channel_11 || \
    DMA_Channel_ADCSEQ3)) || \
  ((CHANNEL) == (DMA_Channel_12 || \
    DMA_Channel_ADCSEQ4)) || \
  ((CHANNEL) == (DMA_Channel_13 || \
    DMA_Channel_ADCSEQ5)) || \
  ((CHANNEL) == (DMA_Channel_14 || \
    DMA_Channel_ADCSEQ6)) || \
  ((CHANNEL) == (DMA_Channel_15 || \
    DMA_Channel_ADCSEQ7)) || \
  ((CHANNEL) == (DMA_Channel_16 || \
    DMA_Channel_SPI0_TX)) || \
  ((CHANNEL) == (DMA_Channel_17 || \
    DMA_Channel_SPI1_TX)) || \
  ((CHANNEL) == (DMA_Channel_18 || \
    DMA_Channel_SPI2_TX)) || \
  ((CHANNEL) == (DMA_Channel_10 || \
    DMA_Channel_SPI3_TX)) || \
  ((CHANNEL) == (DMA_Channel_20 || \
    DMA_Channel_SPI0_RX)) || \
  ((CHANNEL) == (DMA_Channel_21 || \
    DMA_Channel_SPI1_RX)) || \
  ((CHANNEL) == (DMA_Channel_22 || \
    DMA_Channel_SPI2_RX)) || \
  ((CHANNEL) == (DMA_Channel_23 || \
    DMA_Channel_SPI3_RX))))
```

Макрос проверки маски канала при работе с каналами по отдельности.

См. определение в файле `niietcm4_dma.h` строка 166

Используется в `DMA_WaitOnReqStatus()`.

8.10.3 Перечисления

8.10.3.1 `enum DMA_ArbitrationRate_TypeDef`

Выбор количества передач до выполнения переарбитрации.

Элементы перечислений

`DMA_ArbitrationRate_1` Переарбитрация каждую передачу DMA
`DMA_ArbitrationRate_2` Переарбитрация каждые 2 передачи DMA
`DMA_ArbitrationRate_4` Переарбитрация каждые 4 передачи DMA

DMA_ArbitrationRate_8 Переарбитрация каждые 8 передач DMA
 DMA_ArbitrationRate_16 Переарбитрация каждые 16 передач DMA
 DMA_ArbitrationRate_32 Переарбитрация каждые 32 передачи DMA
 DMA_ArbitrationRate_64 Переарбитрация каждые 64 передачи DMA
 DMA_ArbitrationRate_128 Переарбитрация каждые 128 передач DMA
 DMA_ArbitrationRate_256 Переарбитрация каждые 256 передач DMA
 DMA_ArbitrationRate_512 Переарбитрация каждые 512 передач DMA
 DMA_ArbitrationRate_1024 Переарбитрация каждые 1024 передачи DMA

См. определение в файле niietcm4_dma.h строка 297

8.10.3.2 enum DMA_DataInc_TypeDef

Шаг инкремента адреса источника при чтении или приемника при записи

Элементы перечислений

DMA_DataInc_8 Инкремент данных 8 бит
 DMA_DataInc_16 Инкремент данных 16 бит
 DMA_DataInc_32 Инкремент данных 32 бит
 DMA_DataInc_Disable Инкремент отсутствует

См. определение в файле niietcm4_dma.h строка 362

8.10.3.3 enum DMA_DataSize_TypeDef

Разрядность данных источника или приемника

Элементы перечислений

DMA_DataSize_8 Разрядность данных 8 бит
 DMA_DataSize_16 Разрядность данных 16 бит
 DMA_DataSize_32 Разрядность данных 32 бит

См. определение в файле niietcm4_dma.h строка 343

8.10.3.4 enum DMA_Mode_TypeDef

Выбор режима работы DMA.

Элементы перечислений

DMA_Mode_Disable Неактивное состояние
 DMA_Mode_Basic Основной режим передачи
 DMA_Mode_AutoReq Режим передачи с авто-запросом
 DMA_Mode_PingPong Режим передачи "пинг-понг"
 DMA_Mode_PrmMemScatGath Работа с памятью в режиме "разборка-сборка" с использованием первичной управляющей структуры
 DMA_Mode_AltMemScatGath Работа с памятью в режиме "разборка-сборка" с использованием альтернативной управляющей структуры
 DMA_Mode_PrmPeriphScatGath Работа с периферией в режиме "разборка-сборка" с использованием первичной управляющей структуры
 DMA_Mode_AltPeriphScatGath Работа с периферией в режиме "разборка-сборка" с использованием альтернативной управляющей структуры

См. определение в файле niietcm4_dma.h строка 268

8.10.3.5 enum DMA_State_TypeDef

Возможные состояния конечного автомата управления контроллером DMA.

Элементы перечислений

DMA_State_Free В покое.
 DMA_State_ReadConfigData Чтение управляющих данных канала.
 DMA_State_ReadSrcDataEndPtr Чтение указателя конца данных источника.
 DMA_State_ReadDstDataEndPtr Чтение указателя конца данных приемника.
 DMA_State_ReadSrcData Чтение данных источника.
 DMA_State_WriteDstData Запись данных в приемник.
 DMA_State_WaitReq Ожидание запроса на выполнение прямого доступа.
 DMA_State_WriteConfigData Запись управляющих данных канала.
 DMA_State_Pause Приостановлен.
 DMA_State_Done Выполнен.
 DMA_State_PeriphScatGath Работа с периферией в режиме "разборка-сборка".

См. определение в файле niietcm4_dma.h строка 440

8.10.4 Функции

8.10.4.1 void DMA_BasePtrConfig (uint32_t BasePtr)

Установка базового адреса управляющих каналов.

Аргументы

BasePtr	Значение базового адреса.
---------	---------------------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 231

8.10.4.2 void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)

Деинициализация канала DMA.

Аргументы

DMA_Channel	Указатель на структуру типа DMA_Channel_TypeDef , которая содержит конфигурационную информацию канала.
-------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 87

Перекрестные ссылки [DMA_Channel_TypeDef::CHANNEL_CFG](#), [DMA_Channel_TypeDef::DST←_DATA_END](#) и [DMA_Channel_TypeDef::SRC_DATA_END](#).

8.10.4.3 void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)

Активация каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 373

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.10.4.4 void DMA_ChannelInit (DMA_Channel_TypeDef * DMA_Channel,
DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Инициализация канала DMA.

Аргументы

DMA_Channel	Непосредственно сама структура канала.
DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которая содержит конфигурационную информацию канала.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 102

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Channel_TypeDef::CHANNEL_CFG_bit, _CHANNEL_CFG_bits::CYCLE_CTRL, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal, DMA_Channel_TypeDef::DST_DATA_END, _CHANNEL_CFG_bits::DST_INC, _CHANNEL_CFG_bits::DST_PROT_BUFFERABLE, _CHANNEL_CFG_bits::DST_PROT_CACHEABLE, _CHANNEL_CFG_bits::DST_PROT_PRIVILEGED, _CHANNEL_CFG_bits::DST_SIZE, IS_DMA_ARBITRATION_RATE, IS_DMA_DATA_INC, IS_DMA_DATA_SIZE, IS_DMA_MODE, IS_DMA_TRANSFERS_TOTAL, IS_FUNCTIONAL_STATE, _CHANNEL_CFG_bits::N_MINUS_1, _CHANNEL_CFG_bits::NEXT_USEBURST, DMA_Protect_TypeDef::PRIVILEGED, _CHANNEL_CFG_bits::R_POWER, DMA_Channel_TypeDef::SRC_DATA_END, _CHANNEL_CFG_bits::SRC_INC, _CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE, _CHANNEL_CFG_bits::SRC_PROT_CACHEABLE, _CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED и _CHANNEL_CFG_bits::SRC_SIZE.

8.10.4.5 void DMA_ChannelStructInit (DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

Аргументы

DMA_ChannelInitStruct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которую необходимо проинициализировать.
-----------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 146

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHABLE, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ArbitrationRate_1, DMA_DataInc_Disable, DMA_DataSize_8, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_Mode_Disable, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal и DMA_Protect_TypeDef::PRIVELGED.

8.10.4.6 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 524

8.10.4.7 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 174

8.10.4.8 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

Status	Одно из значений OperationStatus: <ul style="list-style-type: none"> • OK - ошибок не было; • ERROR - произошла ошибка.
--------	---

См. определение в файле niietcm4_dma.c строка 503

8.10.4.9 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
-------------	--

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.10.4.10 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

Прежде чем инициализировать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать базовый адрес управляющей структуры с помощью [DMA_BasePtrConfig](#).

Аргументы

DMA_InitStruct	Указатель на структуру типа DMA_Init_TypeDef , которая содержит конфигурационную информацию.
----------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 195

Перекрестные ссылки DMA_Init_TypeDef::DMA_Channel, DMA_Init_TypeDef::DMA_ChannelEnable, DMA_ChannelEnableCmd(), DMA_Init_TypeDef::DMA_HighPriority, DMA_HighPriorityCmd(), DMA_Init_TypeDef::DMA_PrmAlt, DMA_PrmAltCmd(), DMA_Init_TypeDef::DMA_Protection, DMA_ProtectionConfig(), DMA_Init_TypeDef::DMA_ReqMask, DMA_ReqMaskCmd(), DMA_Init_TypeDef::DMA_UseBurst и DMA_UseBurstCmd().

8.10.4.11 void DMA_MasterEnableCmd (FunctionalState State)

Разрешения работы контроллера DMA.

Внимание

Прежде чем включать DMA, необходимо проинициализировать каналы с помощью [DMA_ChannelInit](#) и сконфигурировать контроллер DMA через функцию инициализации [DMA_Init](#) или вручную - [Конфигурация контроллера DMA](#).

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 287

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.10.4.12 FunctionalState DMA_MasterEnableStatus ()

Состояние контроллера DMA.

Возвращаемые значения

Status	Текущее состояние контроллера DMA.
--------	------------------------------------

См. определение в файле niietcm4_dma.c строка 455

8.10.4.13 void DMA_PrmAltCmd (uint32_t DMA_Channel, FunctionalState State)

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 397

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.10.4.14 void DMA_ProtectionConfig (DMA_Protect_TypeDef * DMA_Protection)

Управление защитой шины при обращении DMA к управляющим данным.

Аргументы

DMA_Protection	Структура, содержащая конфигурацию защиты. Параметр принимает структуру типа DMA_Protect_TypeDef .
----------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 243

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, IS_FUNCTIONAL_STATE и DMA_Protect_TypeDef::PRIVELGED.

Используется в DMA_Init().

8.10.4.15 void DMA_ReqMaskCmd (uint32_t DMA_Channel, FunctionalState State)

Маскирование каналов DMA.

Внимание

По маскированным каналам игнорируются запросы на передачи.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
-------------	--

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 349

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

8.10.4.16 `DMA_State_TypeDef DMA_StateStatus ()`

Доступ к текущему конечного автомата контроллера DMA.

Возвращаемые значения

State	Текущее состояние конечного автомата.
-------	---------------------------------------

См. определение в файле `niietcm4_dma.c` строка 441

8.10.4.17 `void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)`

Заполнение каждого члена структуры `DMA_InitStruct` значениями по умолчанию.

Аргументы

<code>DMA_InitStruct</code>	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
-----------------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 212

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHEABLE`, `DMA_Init_TypeDef::DMA_Channel`, `DMA_Init_TypeDef::DMA_ChannelEnable`, `DMA_Init_TypeDef::DMA_HighPriority`, `DMA_Init_TypeDef::DMA_PrmAlt`, `DMA_Init_TypeDef::DMA_Protection`, `DMA_Init_TypeDef::DMA_ReqMask`, `DMA_Init_TypeDef::DMA_UseBurst` и `DMA_Protect_TypeDef::PRIVELGED`.

8.10.4.18 `void DMA_SWRequestCmd (uint32_t DMA_Channel)`

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

<code>DMA_Channel</code>	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
--------------------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_dma.c` строка 308

Перекрестные ссылки `IS_DMA_CHANNEL`.

8.10.4.19 `void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка пакетного обмена каналов DMA.

Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 324

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.10.4.20 FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

Status	Одно из значений FunctionalState: <ul style="list-style-type: none"> • ENABLE - поддерживаются SREQ (как и блочные BREQ); • DISABLE - поддерживаются только блочные запросы BREQ.
--------	---

См. определение в файле niietcm4_dma.c строка 478

Перекрестные ссылки IS_GET_DMA_CHANNEL.

8.11 Файл niietcm4_extmem.c

Файл содержит реализацию всех функции для работы с интерфейсом внешней памяти.

```
#include "niietcm4_extmem.h"
```

Макросы

- #define [EXT_MEM_CFG_Reset_Value](#) ((uint32_t)0x80000007)

Функции

- void [EXTMEM_Init](#) ([EXTMEM_Init_TypeDef](#) *EXTMEM_InitStruct)
Инициализирует внешнюю память согласно параметрам структуры EXTMEM_InitStruct.
- void [EXTMEM_StructInit](#) ([EXTMEM_Init_TypeDef](#) *EXTMEM_InitStruct)
Заполнение каждого члена структуры EXTMEM_InitStruct значениями по умолчанию.
- void [EXTMEM_DeInit](#) ()
Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.

8.11.1 Подробное описание

Файл содержит реализацию всех функции для работы с интерфейсом внешней памяти.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

08.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.11.2 Макросы

8.11.2.1 `#define EXT_MEM_CFG_Reset_Value ((uint32_t)0x80000007)`

Значение по сбросу регистра `EXT_MEM_CFG`

См. определение в файле `niietcm4_extmem.c` строка 64

Используется в `EXTMEM_DeInit()`.

8.11.3 Функции

8.11.3.1 `void EXTMEM_DeInit ()`

Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_extmem.c` строка 121

Перекрестные ссылки `EXT_MEM_CFG_Reset_Value`.

8.11.3.2 `void EXTMEM_Init (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)`

Инициализирует внешнюю память согласно параметрам структуры `EXTMEM_InitStruct`.

Аргументы

EXTMEM_↔ InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 85

Перекрестные ссылки IS_EXTMEM_CE_MASK, IS_EXTMEM_READ_WAITSTATE, IS_EXTMEM_RW_WAITSTATE, IS_EXTMEM_WIDTH и IS_EXTMEM_WRITE_WAITSTATE.

8.11.3.3 void EXTMEM_StructInit (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Заполнение каждого члена структуры EXTMEM_InitStruct значениями по умолчанию.

Аргументы

EXTMEM_↔ InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 107

Перекрестные ссылки EXTMEM_Init_TypeDef::CEMask, EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState, EXTMEM_ReadWaitState_8, EXTMEM_Init_TypeDef::EXTMEM_RWWaitState, EXTMEM_RWWaitState_1, EXTMEM_Init_TypeDef::EXTMEM_Width, EXTMEM_Width_↔16bit, EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState и EXTMEM_WriteWaitState_1.

8.12 Файл niietcm4_extmem.h

Файл содержит все прототипы функций для интерфейса внешней памяти.

```
#include "niietcm4.h"
```

Структуры данных

- struct [EXTMEM_Init_TypeDef](#)
Структура инициализации внешней памяти.

Макросы

- #define [EXTMEM_CEMask_Addr_11](#) ((uint32_t)0x0001)
- #define [EXTMEM_CEMask_Addr_12](#) ((uint32_t)0x0002)
- #define [EXTMEM_CEMask_Addr_13](#) ((uint32_t)0x0004)
- #define [EXTMEM_CEMask_Addr_14](#) ((uint32_t)0x0008)
- #define [EXTMEM_CEMask_Addr_15](#) ((uint32_t)0x0010)
- #define [EXTMEM_CEMask_Addr_16](#) ((uint32_t)0x0020)
- #define [EXTMEM_CEMask_Addr_17](#) ((uint32_t)0x0040)
- #define [EXTMEM_CEMask_Addr_18](#) ((uint32_t)0x0080)
- #define [EXTMEM_CEMask_Addr_19](#) ((uint32_t)0x0100)
- #define [EXTMEM_CEMask_Addr_11_19](#) ((uint32_t)0x01FF)

- `#define IS_EXTMEM_CE_MASK(CE_MASK) (((CE_MASK) & ((uint32_t)0xFFFFFE00)) == ((uint32_t)0x00))`
Макрос проверки соответствия маски адреса разрешенному диапазону.
- `#define IS_EXTMEM_WIDTH(WIDTH)`
Макрос проверки аргументов типа `EXTMEM_Width_TypeDef`.
- `#define IS_EXTMEM_RW_WAITSTATE(WAITSTATE)`
Макрос проверки аргументов типа `EXTMEM_RWWaitState_TypeDef`.
- `#define IS_EXTMEM_WRITE_WAITSTATE(WAITSTATE)`
Макрос проверки аргументов типа `EXTMEM_WriteWaitState_TypeDef`.
- `#define IS_EXTMEM_READ_WAITSTATE(WAITSTATE)`
Макрос проверки аргументов типа `EXTMEM_ReadWaitState_TypeDef`.

Перечисления

- `enum EXTMEM_Width_TypeDef { EXTMEM_Width_8bit, EXTMEM_Width_16bit }`
Разрядность контроллера внешней памяти.
- `enum EXTMEM_RWWaitState_TypeDef { EXTMEM_RWWaitState_1, EXTMEM_RWWaitState_2, EXTMEM_RWWaitState_3, EXTMEM_RWWaitState_4, EXTMEM_RWWaitState_5, EXTMEM_RWWaitState_6, EXTMEM_RWWaitState_7, EXTMEM_RWWaitState_8 }`
Длительность цикла переключения шины в системных тактах.
- `enum EXTMEM_WriteWaitState_TypeDef { EXTMEM_WriteWaitState_1, EXTMEM_WriteWaitState_2, EXTMEM_WriteWaitState_3, EXTMEM_WriteWaitState_4, EXTMEM_WriteWaitState_5, EXTMEM_WriteWaitState_6, EXTMEM_WriteWaitState_7, EXTMEM_WriteWaitState_8 }`
Длительность цикла записи слова данных в системных тактах.
- `enum EXTMEM_ReadWaitState_TypeDef { EXTMEM_ReadWaitState_1, EXTMEM_ReadWaitState_2, EXTMEM_ReadWaitState_3, EXTMEM_ReadWaitState_4, EXTMEM_ReadWaitState_5, EXTMEM_ReadWaitState_6, EXTMEM_ReadWaitState_7, EXTMEM_ReadWaitState_8 }`
Длительность цикла чтения слова данных в системных тактах.

Функции

- `void EXTMEM_DeInit ()`
Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.
- `void EXTMEM_Init (EXTMEM_Init_TypeDef *EXTMEM_InitStruct)`
Инициализирует внешнюю память согласно параметрам структуры `EXTMEM_InitStruct`.
- `void EXTMEM_StructInit (EXTMEM_Init_TypeDef *EXTMEM_InitStruct)`
Заполнение каждого члена структуры `EXTMEM_InitStruct` значениями по умолчанию.

8.12.1 Подробное описание

Файл содержит все прототипы функций для интерфейса внешней памяти.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

08.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.12.2 Макросы

8.12.2.1 `#define EXTMEM_CEMask_Addr_11 ((uint32_t)0x0001)`

Маска бита 11 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 56

8.12.2.2 `#define EXTMEM_CEMask_Addr_11_19 ((uint32_t)0x01FF)`

Маски [19:11] битов адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 65

8.12.2.3 `#define EXTMEM_CEMask_Addr_12 ((uint32_t)0x0002)`

Маска бита 12 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 57

8.12.2.4 `#define EXTMEM_CEMask_Addr_13 ((uint32_t)0x0004)`

Маска бита 13 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 58

8.12.2.5 `#define EXTMEM_CEMask_Addr_14 ((uint32_t)0x0008)`

Маска бита 14 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 59

8.12.2.6 `#define EXTMEM_CEMask_Addr_15 ((uint32_t)0x0010)`

Маска бита 15 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 60

8.12.2.7 `#define EXTMEM_CEMask_Addr_16 ((uint32_t)0x0020)`

Маска бита 16 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 61

8.12.2.8 `#define EXTMEM_CEMask_Addr_17 ((uint32_t)0x0040)`

Маска бита 17 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 62

8.12.2.9 `#define EXTMEM_CEMask_Addr_18 ((uint32_t)0x0080)`

Маска бита 18 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 63

8.12.2.10 `#define EXTMEM_CEMask_Addr_19 ((uint32_t)0x0100)`

Маска бита 19 адреса контроллера внешней памяти.

См. определение в файле `niietcm4_extmem.h` строка 64

8.12.2.11 `#define IS_EXTMEM_READ_WAITSTATE(WAITSTATE)`

Макроопределение:

```
(((WAITSTATE) == EXTMEM_ReadWaitState_1) || \
  EXTMEM_ReadWaitState_2) || \
  ((WAITSTATE) ==
  EXTMEM_ReadWaitState_3) || \
  ((WAITSTATE) ==
  EXTMEM_ReadWaitState_4) || \
  ((WAITSTATE) ==
  EXTMEM_ReadWaitState_5) || \
  ((WAITSTATE) ==
  EXTMEM_ReadWaitState_6) || \
  ((WAITSTATE) ==
  EXTMEM_ReadWaitState_7) || \
  ((WAITSTATE) ==
  EXTMEM_ReadWaitState_8))
```

Макрос проверки аргументов типа `EXTMEM_ReadWaitState_TypeDef`.

См. определение в файле `niietcm4_extmem.h` строка 180

Используется в `EXTMEM_Init()`.

8.12.2.12 `#define IS_EXTMEM_RW_WAITSTATE(WAITSTATE)`

Макроопределение:

```
(((WAITSTATE) == EXTMEM_RWWaitState_1) || \
  ((WAITSTATE) == EXTMEM_RWWaitState_2) || \
  ((WAITSTATE) == EXTMEM_RWWaitState_3) || \
  ((WAITSTATE) == EXTMEM_RWWaitState_4) || \
```

```
((WAITSTATE) == EXTMEM_RWWaitState_5) || \
((WAITSTATE) == EXTMEM_RWWaitState_6) || \
((WAITSTATE) == EXTMEM_RWWaitState_7) || \
((WAITSTATE) == EXTMEM_RWWaitState_8))
```

Макрос проверки аргументов типа [EXTMEM_RWWaitState_TypeDef](#).

См. определение в файле niietcm4_extmem.h строка 122

Используется в EXTMEM_Init().

8.12.2.13 #define IS_EXTMEM_WIDTH(WIDTH)

Макроопределение:

```
((WIDTH) == EXTMEM_Width_8bit) || \
((WIDTH) == EXTMEM_Width_16bit))
```

Макрос проверки аргументов типа [EXTMEM_Width_TypeDef](#).

См. определение в файле niietcm4_extmem.h строка 99

Используется в EXTMEM_Init().

8.12.2.14 #define IS_EXTMEM_WRITE_WAITSTATE(WAITSTATE)

Макроопределение:

```
((WAITSTATE) == EXTMEM_WriteWaitState_1) || \
((WAITSTATE) == EXTMEM_WriteWaitState_2) || \
((WAITSTATE) == EXTMEM_WriteWaitState_3) || \
((WAITSTATE) == EXTMEM_WriteWaitState_4) || \
((WAITSTATE) == EXTMEM_WriteWaitState_5) || \
((WAITSTATE) == EXTMEM_WriteWaitState_6) || \
((WAITSTATE) == EXTMEM_WriteWaitState_7) || \
((WAITSTATE) == EXTMEM_WriteWaitState_8))
```

Макрос проверки аргументов типа [EXTMEM_WriteWaitState_TypeDef](#).

См. определение в файле niietcm4_extmem.h строка 151

Используется в EXTMEM_Init().

8.12.3 Перечисления

8.12.3.1 enum EXTMEM_ReadWaitState_TypeDef

Длительность цикла чтения слова данных в системных тактах.

Элементы перечислений

```
EXTMEM_ReadWaitState_1 1 цикл ожидания.
EXTMEM_ReadWaitState_2 2 цикла ожидания.
EXTMEM_ReadWaitState_3 3 цикла ожидания.
EXTMEM_ReadWaitState_4 4 цикла ожидания.
EXTMEM_ReadWaitState_5 5 циклов ожидания.
EXTMEM_ReadWaitState_6 6 циклов ожидания.
```

EXTMEM_ReadWaitState_7 7 циклов ожидания.

EXTMEM_ReadWaitState_8 8 циклов ожидания.

См. определение в файле niietcm4_extmem.h строка 164

8.12.3.2 enum EXTMEM_RWWaitState_TypeDef

Длительность цикла переключения шины в системных тактах.

Элементы перечислений

EXTMEM_RWWaitState_1 1 цикл ожидания.

EXTMEM_RWWaitState_2 2 цикла ожидания.

EXTMEM_RWWaitState_3 3 цикла ожидания.

EXTMEM_RWWaitState_4 4 цикла ожидания.

EXTMEM_RWWaitState_5 5 циклов ожидания.

EXTMEM_RWWaitState_6 6 циклов ожидания.

EXTMEM_RWWaitState_7 7 циклов ожидания.

EXTMEM_RWWaitState_8 8 циклов ожидания.

См. определение в файле niietcm4_extmem.h строка 106

8.12.3.3 enum EXTMEM_Width_TypeDef

Разрядность контроллера внешней памяти.

Элементы перечислений

EXTMEM_Width_8bit 8-разрядный режим работы.

EXTMEM_Width_16bit 16-разрядный режим работы.

См. определение в файле niietcm4_extmem.h строка 89

8.12.3.4 enum EXTMEM_WriteWaitState_TypeDef

Длительность цикла записи слова данных в системных тактах.

Элементы перечислений

EXTMEM_WriteWaitState_1 1 цикл ожидания.

EXTMEM_WriteWaitState_2 2 цикла ожидания.

EXTMEM_WriteWaitState_3 3 цикла ожидания.

EXTMEM_WriteWaitState_4 4 цикла ожидания.

EXTMEM_WriteWaitState_5 5 циклов ожидания.

EXTMEM_WriteWaitState_6 6 циклов ожидания.

EXTMEM_WriteWaitState_7 7 циклов ожидания.

EXTMEM_WriteWaitState_8 8 циклов ожидания.

См. определение в файле niietcm4_extmem.h строка 135

8.12.4 Функции

8.12.4.1 void EXTMEM_DeInit ()

Устанавливает все регистры контроллера внешней памяти значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 121

Перекрестные ссылки EXT_MEM_CFG_Reset_Value.

8.12.4.2 void EXTMEM_Init (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Инициализирует внешнюю память согласно параметрам структуры EXTMEM_InitStruct.

Аргументы

EXTMEM_InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
-------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 85

Перекрестные ссылки IS_EXTMEM_CE_MASK, IS_EXTMEM_READ_WAITSTATE, IS_EXTMEM_RW_WAITSTATE, IS_EXTMEM_WIDTH и IS_EXTMEM_WRITE_WAITSTATE.

8.12.4.3 void EXTMEM_StructInit (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Заполнение каждого члена структуры EXTMEM_InitStruct значениями по умолчанию.

Аргументы

EXTMEM_InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
-------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 107

Перекрестные ссылки EXTMEM_Init_TypeDef::CEMask, EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState, EXTMEM_ReadWaitState_8, EXTMEM_Init_TypeDef::EXTMEM_RWWaitState, EXTMEM_RWWaitState_1, EXTMEM_Init_TypeDef::EXTMEM_Width, EXTMEM_Width_16bit, EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState и EXTMEM_WriteWaitState_1.

8.13 Файл niietcm4_gpio.c

Файл содержит реализацию всех функции для работы с модулями GPIO.

```
#include "niietcm4_gpio.h"
```

Макросы

- #define [GPIO_DATAOUT_Reset_Value](#) ((uint32_t)0x00000000)
- #define [GPIO_GPIODEN0_Reset_Value](#) ((uint32_t)0x00020062)
- #define [GPIO_GPIODEN1_Reset_Value](#) ((uint32_t)0x08000000)
- #define [GPIO_GPIODEN2_Reset_Value](#) ((uint32_t)0x00000400)
- #define [GPIO_GPIODEN3_Reset_Value](#) ((uint32_t)0x00000000)
- #define [GPIO_GPIOODCTLx_Reset_Value](#) ((uint32_t)0x00000000)

- `#define GPIO_GPIOODSCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOPCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOSEx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQEx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQMx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOPCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQPx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_Regs_A_C_E_G_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_B_D_F_H_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOA_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOB_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOC_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOD_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOE_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOF_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOG_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

Функции

- `void GPIO_DeInit (NT_GPIO_TypeDef *GPIOx)`
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- `void GPIO_AltFuncConfig (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc)`
- `void GPIO_Init (NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)`
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- `void GPIO_StructInit (GPIO_Init_TypeDef *GPIO_InitStruct)`
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.
- `uint32_t GPIO_ReadBit (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Чтение состояния выбранного пина.
- `uint32_t GPIO_Read (NT_GPIO_TypeDef *GPIOx)`
Чтение состояния выбранного порта GPIOx.
- `uint32_t GPIO_ReadMask (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)`
Чтение состояния выбранного порта GPIOx с использованием маски.
- `void GPIO_WriteBit (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)`
Изменение состояния выбранного пина.
- `void GPIO_Write (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)`
Изменение состояния выбранного порта GPIOx.
- `void GPIO_WriteMask (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)`
Изменение состояния выбранного порта GPIOx с использованием маски.
- `void GPIO_SetBits (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Установка выбранных пинов.
- `void GPIO_ClearBits (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Сброс выбранных пинов.
- `void GPIO_ToggleBits (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Переключение выбранных пинов в противоположное состояние.
- `void GPIO_QualConfig (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)`
Настройка фильтра выбранных пинов.
- `void GPIO_QualCmd (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)`
Включение входных фильтров.

- void `GPIO_SyncCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)
Включение пересинхронизации входов через 2 триггера-защелки.
- void `GPIO_ITConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)
Настройка прерываний пинов.
- void `GPIO_ITCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)
Включение прерываний выбранных пинов.
- void `GPIO_ITStatusClear` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Очистка флагов прерываний выбранных пинов.

8.13.1 Подробное описание

Файл содержит реализацию всех функции для работы с модулями GPIO.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.13.2 Макросы

8.13.2.1 `#define GPIO_DATAOUT_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра DATAOUT

См. определение в файле niietcm4_gpio.c строка 72

Используется в `GPIO_DeInit()`.

8.13.2.2 `#define GPIO_GPIODEN0_Reset_Value ((uint32_t)0x00020062)`

Значение по сбросу регистра GPIODEN0

См. определение в файле `niietcm4_gpio.c` строка 73

Используется в `GPIO_DeInit()`.

8.13.2.3 `#define GPIO_GPIODEN1_Reset_Value ((uint32_t)0x08000000)`

Значение по сбросу регистра GPIODEN1

См. определение в файле `niietcm4_gpio.c` строка 74

Используется в `GPIO_DeInit()`.

8.13.2.4 `#define GPIO_GPIODEN2_Reset_Value ((uint32_t)0x00000400)`

Значение по сбросу регистра GPIODEN2

См. определение в файле `niietcm4_gpio.c` строка 75

Используется в `GPIO_DeInit()`.

8.13.2.5 `#define GPIO_GPIODEN3_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIODEN3

См. определение в файле `niietcm4_gpio.c` строка 76

Используется в `GPIO_DeInit()`.

8.13.2.6 `#define GPIO_GPIOODCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOODCTLx

См. определение в файле `niietcm4_gpio.c` строка 77

Используется в `GPIO_DeInit()`.

8.13.2.7 `#define GPIO_GPIOODSCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOODSCTLx

См. определение в файле `niietcm4_gpio.c` строка 78

Используется в `GPIO_DeInit()`.

8.13.2.8 `#define GPIO_GPIOPCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOPCLTx

См. определение в файле `niietcm4_gpio.c` строка 83

Используется в `GPIO_DeInit()`.

8.13.2.9 `#define GPIO_GPIOPUCTLx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOPUCTLx

См. определение в файле `niietcm4_gpio.c` строка 79

Используется в `GPIO_DeInit()`.

8.13.2.10 `#define GPIO_GPIOQEx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOQEx

См. определение в файле niietcm4_gpio.c строка 81

Используется в GPIO_DeInit().

8.13.2.11 `#define GPIO_GPIOQMx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOQMx

См. определение в файле niietcm4_gpio.c строка 82

Используется в GPIO_DeInit().

8.13.2.12 `#define GPIO_GPIOQPx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOQPx

См. определение в файле niietcm4_gpio.c строка 84

Используется в GPIO_DeInit().

8.13.2.13 `#define GPIO_GPIOSEx_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра GPIOSEx

См. определение в файле niietcm4_gpio.c строка 80

Используется в GPIO_DeInit().

8.13.2.14 `#define GPIO_Regs_A_C_E_G_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для нечетных портов

См. определение в файле niietcm4_gpio.c строка 94

Используется в GPIO_DeInit().

8.13.2.15 `#define GPIO_Regs_B_D_F_H_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для четных портов

См. определение в файле niietcm4_gpio.c строка 95

Используется в GPIO_DeInit().

8.13.2.16 `#define GPIO_Regs_GPIOA_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOA

См. определение в файле niietcm4_gpio.c строка 96

8.13.2.17 `#define GPIO_Regs_GPIOB_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOB

См. определение в файле niietcm4_gpio.c строка 97

8.13.2.18 `#define GPIO_Regs_GPIOC_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOC

См. определение в файле `niietcm4_gpio.c` строка 98

8.13.2.19 `#define GPIO_Regs_GPIOD_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOD

См. определение в файле `niietcm4_gpio.c` строка 99

8.13.2.20 `#define GPIO_Regs_GPIOE_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOE

См. определение в файле `niietcm4_gpio.c` строка 100

8.13.2.21 `#define GPIO_Regs_GPIOF_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOF

См. определение в файле `niietcm4_gpio.c` строка 101

8.13.2.22 `#define GPIO_Regs_GPIOG_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOG

См. определение в файле `niietcm4_gpio.c` строка 102

8.13.2.23 `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOH

См. определение в файле `niietcm4_gpio.c` строка 103

8.13.3 Функции

8.13.3.1 `void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 626

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.13.3.2 `void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)`

Устанавливает все регистры выбранного GPIOx значениями по умолчанию.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 123

Перекрестные ссылки GPIO_DATAOUT_Reset_Value, GPIO_GPIODEN0_Reset_Value, GPIO_GPIODEN1_Reset_Value, GPIO_GPIODEN2_Reset_Value, GPIO_GPIODEN3_Reset_Value, GPIO_GPIOODCTLx_Reset_Value, GPIO_GPIOODSCTLx_Reset_Value, GPIO_GPIOPCTLx_Reset_Value, GPIO_GPIOPUCTLx_Reset_Value, GPIO_GPIOQEx_Reset_Value, GPIO_GPIOQMx_Reset_Value, GPIO_GPIOQPx_Reset_Value, GPIO_GPIOSEx_Reset_Value, GPIO_Regs_A_C_E_G_Mask, GPIO_Regs_B_D_F_H_Mask и IS_GPIO_ALL_PERIPH.

8.13.3.3 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 331

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_In, GPIO_Dir_Out, GPIO_Init_TypeDef::GPIO_Load, GPIO_Load_16mA, GPIO_Load_8mA, GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Mode_IO, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_Dis, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_OutMode, GPIO_OutMode_OD, GPIO_OutMode_PP, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Init_TypeDef::GPIO_PullUp, GPIO_PullUp_Dis, GPIO_PullUp_En, IS_GPIO_ALL_PERIPH, IS_GPIO_ALT_FUNC, IS_GPIO_DIR, IS_GPIO_LOAD, IS_GPIO_MODE, IS_GPIO_OUT, IS_GPIO_OUT_MODE, IS_GPIO_PIN и IS_GPIO_PULLUP.

Используется в RCC_SysClkDiv2Out().

8.13.3.4 void GPIO_ITCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 913

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.13.3.5 `void GPIO_ITConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)`

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 876

Перекрестные ссылки `GPIO_IntPol_Neg`, `GPIO_IntPol_Pos`, `GPIO_IntType_Edge`, `GPIO_IntType_Level`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_INT_POL`, `IS_GPIO_INT_TYPE` и `IS_GPIO_PIN`.

8.13.3.6 `void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 938

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.13.3.7 `void GPIO_QualCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)`

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.13.3.8 void GPIO_QualConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 664

Перекрестные ссылки GPIO_QualMode_3sample, GPIO_QualMode_6sample, IS_GPIO_ALL_PERIPH, IS_GPIO_PIN, IS_GPIO_QUAL_MODE и IS_GPIO_QUAL_PERIOD.

8.13.3.9 uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx	
-----------------------	--

См. определение в файле niietcm4_gpio.c строка 503

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.13.3.10 uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина	
------------------------------	--

См. определение в файле `niietcm4_gpio.c` строка 477

Перекрестные ссылки `Bit_CLEAR`, `Bit_SET`, `IS_GET_GPIO_PIN` и `IS_GPIO_ALL_PERIPH`.

8.13.3.11 `uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)`

Чтение состояния выбранного порта `GPIOx` с использованием маски.

Аргументы

<code>GPIOx</code>	Выбор порта, где <code>x</code> лежит в диапазоне <code>A..H</code> .
<code>MaskVal</code>	Значение маски чтения.

Возвращаемые значения

Состояние порта <code>GPIOx</code> с учетом маски	
--	--

См. определение в файле `niietcm4_gpio.c` строка 518

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

8.13.3.12 `void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Установка выбранных пинов.

Аргументы

<code>GPIOx</code>	Выбор порта, где <code>x</code> лежит в диапазоне <code>A..H</code> .
<code>GPIO_Pin</code>	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где <code>x</code> лежит в диапазоне <code>0..15</code> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 609

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.13.3.13 `void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)`

Заполнение каждого члена структуры `GPIO_InitStruct` значениями по умолчанию.

Аргументы

<code>GPIO_InitStruct</code>	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
------------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 456

Перекрестные ссылки `GPIO_Init_TypeDef::GPIO_AltFunc`, `GPIO_AltFunc_1`, `GPIO_Init_TypeDef::GPIO_Dir`, `GPIO_Dir_In`, `GPIO_Init_TypeDef::GPIO_Load`, `GPIO_Load_8mA`, `GPIO_Init_TypeDef::GPIO_Mode`, `GPIO_Mode_IO`, `GPIO_Init_TypeDef::GPIO_Out`, `GPIO_Out_Dis`, `GPIO_Init_TypeDef::GPIO_OutMode`, `GPIO_OutMode_PP`, `GPIO_Init_TypeDef::GPIO_Pin`, `GPIO_Init_TypeDef::GPIO_Speed`, `GPIO_Speed_2MHz`.

IO_Pin_All, GPIO_Init_TypeDef::GPIO_PullUp и GPIO_PullUp_Dis.

Используется в RCC_SysClkDiv2Out().

8.13.3.14 void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение пересинхронизации входов через 2 триггера-зашелки.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 814

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.13.3.15 void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Переключение выбранных пинов в противоположное состояние.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 643

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.13.3.16 void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.13.3.17 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 546

Перекрестные ссылки `Bit_CLEAR`, `Bit_SET`, `IS_GET_GPIO_PIN`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_BIT_ACTION`.

8.13.3.18 `void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)`

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 586

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

8.14 Файл `niietcm4_gpio.h`

Файл содержит все прототипы функций для GPIO.

```
#include "niietcm4.h"
```

Структуры данных

- struct [GPIO_Init_TypeDef](#)
Структура инициализации GPIO.

Макросы

- `#define IS_GPIO_QUAL_PERIOD(PERIOD) (((PERIOD) & ((uint32_t)0xFFFFF00)) == ((uint32_t)0x00))`
Макрос проверки соответствия величины периода фильтрации разрешенному диапазону.
- `#define IS_GPIO_BIT_ACTION(ACTION) (((ACTION) == Bit_CLEAR) || ((ACTION) == Bit_SET))`
Макрос проверки аргументов типа [BitAction](#).
- `#define IS_GPIO_DIR(DIR)`
Макрос проверки аргументов типа [GPIO_Dir_TypeDef](#).

- `#define IS_GPIO_MODE(MODE)`
Макрос проверки аргументов типа `GPIO_Mode_TypeDef`.
- `#define IS_GPIO_INT_TYPE(INT_TYPE)`
Макрос проверки аргументов типа `GPIO_IntType_TypeDef`.
- `#define IS_GPIO_INT_POL(INT_POL)`
Макрос проверки аргументов типа `GPIO_IntPol_TypeDef`.
- `#define IS_GPIO_OUT(OUT)`
Макрос проверки аргументов типа `GPIO_Out_TypeDef`.
- `#define IS_GPIO_LOAD(LOAD)`
Макрос проверки аргументов типа `GPIO_Load_TypeDef`.
- `#define IS_GPIO_OUT_MODE(OUT_MODE)`
Макрос проверки аргументов типа `GPIO_OutMode_TypeDef`.
- `#define IS_GPIO_PULLUP(PULLUP)`
Макрос проверки аргументов типа `GPIO_PullUp_TypeDef`.
- `#define IS_GPIO_SYNC(SYNC)`
Макрос проверки аргументов типа `GPIO_Sync_TypeDef`.
- `#define IS_GPIO_QUAL(QUAL)`
Макрос проверки аргументов типа `GPIO_Qual_TypeDef`.
- `#define IS_GPIO_QUAL_MODE(QUAL_MODE)`
Макрос проверки аргументов типа `GPIO_QualMode_TypeDef`.
- `#define IS_GPIO_ALT_FUNC(ALT_FUNC)`
Макрос проверки аргументов типа `GPIO_AltFunc_TypeDef`.
- `#define GPIO_Pin_0 ((uint32_t)0x0001)`
- `#define GPIO_Pin_1 ((uint32_t)0x0002)`
- `#define GPIO_Pin_2 ((uint32_t)0x0004)`
- `#define GPIO_Pin_3 ((uint32_t)0x0008)`
- `#define GPIO_Pin_4 ((uint32_t)0x0010)`
- `#define GPIO_Pin_5 ((uint32_t)0x0020)`
- `#define GPIO_Pin_6 ((uint32_t)0x0040)`
- `#define GPIO_Pin_7 ((uint32_t)0x0080)`
- `#define GPIO_Pin_8 ((uint32_t)0x0100)`
- `#define GPIO_Pin_9 ((uint32_t)0x0200)`
- `#define GPIO_Pin_10 ((uint32_t)0x0400)`
- `#define GPIO_Pin_11 ((uint32_t)0x0800)`
- `#define GPIO_Pin_12 ((uint32_t)0x1000)`
- `#define GPIO_Pin_13 ((uint32_t)0x2000)`
- `#define GPIO_Pin_14 ((uint32_t)0x4000)`
- `#define GPIO_Pin_15 ((uint32_t)0x8000)`
- `#define GPIO_Pin_0_3 ((uint32_t)0x000F)`
- `#define GPIO_Pin_4_7 ((uint32_t)0x00F0)`
- `#define GPIO_Pin_8_11 ((uint32_t)0x0F00)`
- `#define GPIO_Pin_12_15 ((uint32_t)0xF000)`
- `#define GPIO_Pin_0_7 ((uint32_t)0x00FF)`
- `#define GPIO_Pin_8_15 ((uint32_t)0xFF00)`
- `#define GPIO_Pin_All ((uint32_t)0xFFFF)`
- `#define IS_GPIO_PIN(PIN) (((PIN) != (uint32_t)0x0000) && (((PIN) & (uint32_t)0xFFFF) <= 0xFFFF))`
Макрос проверки номеров пинов на попадание в допустимый диапазон.
- `#define IS_GET_GPIO_PIN(PIN)`
Макрос проверки номера пина при работе с пинами по отдельности.

Перечисления

- enum `BitAction` { `Bit_CLEAR` = 0, `Bit_SET` }
Тип, определяющий состояния бита.
- enum `GPIO_Dir_TypeDef` { `GPIO_Dir_In`, `GPIO_Dir_Out` }
Выбор направления работы пина.
- enum `GPIO_Mode_TypeDef` { `GPIO_Mode_IO`, `GPIO_Mode_AltFunc` }
Выбор режима работы пина.
- enum `GPIO_IntType_TypeDef` { `GPIO_IntType_Level`, `GPIO_IntType_Edge` }
Выбор события для возникновения прерывания.
- enum `GPIO_IntPol_TypeDef` { `GPIO_IntPol_Neg`, `GPIO_IntPol_Pos` }
Выбор полярности события для возникновения прерывания.
- enum `GPIO_Out_TypeDef` { `GPIO_Out_Dis`, `GPIO_Out_En` }
Включение выхода пина.
- enum `GPIO_Load_TypeDef` { `GPIO_Load_8mA`, `GPIO_Load_16mA` }
Выбор максимальной нагрузочной способности пина.
- enum `GPIO_OutMode_TypeDef` { `GPIO_OutMode_PP`, `GPIO_OutMode_OD` }
Выбор режима работы выходных каскадов.
- enum `GPIO_PullUp_TypeDef` { `GPIO_PullUp_Dis`, `GPIO_PullUp_En` }
Включение подтяжки к питанию.
- enum `GPIO_Sync_TypeDef` { `GPIO_Sync_Dis`, `GPIO_Sync_En` }
Включение режима пересинхронизации входов через 2 триггера-защелки.
- enum `GPIO_Qual_TypeDef` { `GPIO_Qual_Dis`, `GPIO_Qual_En` }
Включение входного фильтра.
- enum `GPIO_QualMode_TypeDef` { `GPIO_QualMode_3sample`, `GPIO_QualMode_6sample` }
Выбор режима работы входного фильтра.
- enum `GPIO_AltFunc_TypeDef` { `GPIO_AltFunc_1`, `GPIO_AltFunc_2`, `GPIO_AltFunc_3` }
Выбор номера альтернативной функции пина.

Функции

- void `GPIO_DeInit` (NT_GPIO_TypeDef *GPIOx)
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- void `GPIO_Init` (NT_GPIO_TypeDef *GPIOx, `GPIO_Init_TypeDef` *GPIO_InitStruct)
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- void `GPIO_StructInit` (`GPIO_Init_TypeDef` *GPIO_InitStruct)
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.
- void `GPIO_AltFuncConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `GPIO_AltFunc_TypeDef` GPIO_AltFunc)
- uint32_t `GPIO_ReadBit` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Чтение состояния выбранного пина.
- uint32_t `GPIO_Read` (NT_GPIO_TypeDef *GPIOx)
Чтение состояния выбранного порта GPIOx.
- uint32_t `GPIO_ReadMask` (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)
Чтение состояния выбранного порта GPIOx с использованием маски.
- void `GPIO_WriteBit` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `BitAction` BitVal)
Изменение состояния выбранного пина.
- void `GPIO_Write` (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx.
- void `GPIO_WriteMask` (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx с использованием маски.

- void `GPIO_SetBits` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Установка выбранных пинов.
- void `GPIO_ClearBits` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Сброс выбранных пинов.
- void `GPIO_ToggleBits` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Переключение выбранных пинов в противоположное состояние.
- void `GPIO_QualConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `GPIO_QualMode_TypeDef` Mode, uint32_t SamplePerod)
Настройка фильтра выбранных пинов.
- void `GPIO_QualCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `FunctionalState` State)
Включение входных фильтров.
- void `GPIO_SyncCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `FunctionalState` State)
Включение пересинхронизации входов через 2 триггера-защелки.
- void `GPIO_ITConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `GPIO_IntType_TypeDef` IntType, `GPIO_IntPol_TypeDef` IntPol)
Настройка прерываний пинов.
- void `GPIO_ITCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `FunctionalState` State)
Включение прерываний выбранных пинов.
- void `GPIO_ITStatusClear` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Очистка флагов прерываний выбранных пинов.

8.14.1 Подробное описание

Файл содержит все прототипы функций для GPIO.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.14.2 Макросы

8.14.2.1 `#define GPIO_Pin_0 ((uint32_t)0x0001)`

Пин 0 выбран.

См. определение в файле `niietcm4_gpio.h` строка 319

Используется в `RCC_SysClkDiv2Out()`.

8.14.2.2 `#define GPIO_Pin_0_3 ((uint32_t)0x000F)`

Пины 0-3 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 335

8.14.2.3 `#define GPIO_Pin_0_7 ((uint32_t)0x00FF)`

Пины 0-7 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 339

8.14.2.4 `#define GPIO_Pin_1 ((uint32_t)0x0002)`

Пин 1 выбран.

См. определение в файле `niietcm4_gpio.h` строка 320

8.14.2.5 `#define GPIO_Pin_10 ((uint32_t)0x0400)`

Пин 10 выбран.

См. определение в файле `niietcm4_gpio.h` строка 329

8.14.2.6 `#define GPIO_Pin_11 ((uint32_t)0x0800)`

Пин 11 выбран.

См. определение в файле `niietcm4_gpio.h` строка 330

8.14.2.7 `#define GPIO_Pin_12 ((uint32_t)0x1000)`

Пин 12 выбран.

См. определение в файле `niietcm4_gpio.h` строка 331

8.14.2.8 `#define GPIO_Pin_12_15 ((uint32_t)0xF000)`

Пины 12-15 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 338

8.14.2.9 `#define GPIO_Pin_13 ((uint32_t)0x2000)`

Пин 13 выбран.

См. определение в файле niietcm4_gpio.h строка 332

8.14.2.10 `#define GPIO_Pin_14 ((uint32_t)0x4000)`

Пин 14 выбран.

См. определение в файле niietcm4_gpio.h строка 333

8.14.2.11 `#define GPIO_Pin_15 ((uint32_t)0x8000)`

Пин 15 выбран.

См. определение в файле niietcm4_gpio.h строка 334

8.14.2.12 `#define GPIO_Pin_2 ((uint32_t)0x0004)`

Пин 2 выбран.

См. определение в файле niietcm4_gpio.h строка 321

8.14.2.13 `#define GPIO_Pin_3 ((uint32_t)0x0008)`

Пин 3 выбран.

См. определение в файле niietcm4_gpio.h строка 322

8.14.2.14 `#define GPIO_Pin_4 ((uint32_t)0x0010)`

Пин 4 выбран.

См. определение в файле niietcm4_gpio.h строка 323

8.14.2.15 `#define GPIO_Pin_4_7 ((uint32_t)0x00F0)`

Пины 4-7 выбраны.

См. определение в файле niietcm4_gpio.h строка 336

8.14.2.16 `#define GPIO_Pin_5 ((uint32_t)0x0020)`

Пин 5 выбран.

См. определение в файле niietcm4_gpio.h строка 324

8.14.2.17 `#define GPIO_Pin_6 ((uint32_t)0x0040)`

Пин 6 выбран.

См. определение в файле niietcm4_gpio.h строка 325

8.14.2.18 `#define GPIO_Pin_7 ((uint32_t)0x0080)`

Пин 7 выбран.

См. определение в файле niietcm4_gpio.h строка 326

8.14.2.19 `#define GPIO_Pin_8 ((uint32_t)0x0100)`

Пин 8 выбран.

См. определение в файле niietcm4_gpio.h строка 327

8.14.2.20 `#define GPIO_Pin_8_11 ((uint32_t)0x0F00)`

Пины 8-11 выбраны.

См. определение в файле niietcm4_gpio.h строка 337

8.14.2.21 `#define GPIO_Pin_8_15 ((uint32_t)0xFF00)`

Пины 8-15 выбраны.

См. определение в файле niietcm4_gpio.h строка 340

8.14.2.22 `#define GPIO_Pin_9 ((uint32_t)0x0200)`

Пин 9 выбран.

См. определение в файле niietcm4_gpio.h строка 328

8.14.2.23 `#define GPIO_Pin_All ((uint32_t)0xFFFF)`

Все пины выбраны.

См. определение в файле niietcm4_gpio.h строка 341

Используется в `GPIO_StructInit()`.

8.14.2.24 `#define IS_GET_GPIO_PIN(PIN)`

Макроопределение:

```
((PIN) == GPIO_Pin_0) || \
    ((PIN) == GPIO_Pin_1) || \
    ((PIN) == GPIO_Pin_2) || \
    ((PIN) == GPIO_Pin_3) || \
    ((PIN) == GPIO_Pin_4) || \
    ((PIN) == GPIO_Pin_5) || \
    ((PIN) == GPIO_Pin_6) || \
    ((PIN) == GPIO_Pin_7) || \
    ((PIN) == GPIO_Pin_8) || \
    ((PIN) == GPIO_Pin_9) || \
    ((PIN) == GPIO_Pin_10) || \
    ((PIN) == GPIO_Pin_11) || \
    ((PIN) == GPIO_Pin_12) || \
    ((PIN) == GPIO_Pin_13) || \
    ((PIN) == GPIO_Pin_14) || \
    ((PIN) == GPIO_Pin_15))
```

Макрос проверки номера пина при работе с пинами по отдельности.

См. определение в файле niietcm4_gpio.h строка 354

Используется в `GPIO_ReadBit()` и `GPIO_WriteBit()`.

8.14.2.25 `#define IS_GPIO_ALT_FUNC(ALT_FUNC)`

Макроопределение:

```
((ALT_FUNC) == GPIO_AltFunc_1) || \
((ALT_FUNC) == GPIO_AltFunc_2) || \
((ALT_FUNC) == GPIO_AltFunc_3))
```

Макрос проверки аргументов типа `GPIO_AltFunc_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 276

Используется в `GPIO_Init()`.

8.14.2.26 `#define IS_GPIO_DIR(DIR)`

Макроопределение:

```
((DIR) == GPIO_Dir_In) || \
((DIR) == GPIO_Dir_Out))
```

Макрос проверки аргументов типа `GPIO_Dir_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 88

Используется в `GPIO_Init()`.

8.14.2.27 `#define IS_GPIO_INT_POL(INT_POL)`

Макроопределение:

```
((INT_POL) == GPIO_IntPol_Neg) || \
((INT_POL) == GPIO_IntPol_Pos))
```

Макрос проверки аргументов типа `GPIO_IntPol_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 139

Используется в `GPIO_ITConfig()`.

8.14.2.28 `#define IS_GPIO_INT_TYPE(INT_TYPE)`

Макроопределение:

```
((INT_TYPE) == GPIO_IntType_Level) || \
((INT_TYPE) == GPIO_IntType_Edge))
```

Макрос проверки аргументов типа `GPIO_IntType_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 122

Используется в `GPIO_ITConfig()`.

8.14.2.29 `#define IS_GPIO_LOAD(LOAD)`

Макроопределение:

```
((LOAD) == GPIO_Load_8mA) || \
((LOAD) == GPIO_Load_16mA))
```

Макрос проверки аргументов типа `GPIO_Load_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 173

Используется в `GPIO_Init()`.

8.14.2.30 `#define IS_GPIO_MODE(MODE)`

Макроопределение:

```
((MODE) == GPIO_Mode_IO) || \
((MODE) == GPIO_Mode_AltFunc))
```

Макрос проверки аргументов типа `GPIO_Mode_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 105

Используется в `GPIO_Init()`.

8.14.2.31 `#define IS_GPIO_OUT(OUT)`

Макроопределение:

```
((OUT) == GPIO_Out_Dis) || \
((OUT) == GPIO_Out_En))
```

Макрос проверки аргументов типа `GPIO_Out_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 156

Используется в `GPIO_Init()`.

8.14.2.32 `#define IS_GPIO_OUT_MODE(OUT_MODE)`

Макроопределение:

```
((OUT_MODE) == GPIO_OutMode_PP) || \
((OUT_MODE) == GPIO_OutMode_OD))
```

Макрос проверки аргументов типа `GPIO_OutMode_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 190

Используется в `GPIO_Init()`.

8.14.2.33 `#define IS_GPIO_PULLUP(PULLUP)`

Макроопределение:

```
((PULLUP) == GPIO_PullUp_Dis) || \
((PULLUP) == GPIO_PullUp_En))
```

Макрос проверки аргументов типа `GPIO_PullUp_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 207

Используется в `GPIO_Init()`.

8.14.2.34 `#define IS_GPIO_QUAL(QUAL)`

Макроопределение:

```
((QUAL) == GPIO_Qual_Dis) || \
((QUAL) == GPIO_Qual_En))
```

Макрос проверки аргументов типа `GPIO_Qual_TypeDef`.

См. определение в файле `niietcm4_gpio.h` строка 241

8.14.2.35 `#define IS_GPIO_QUAL_MODE(QUAL_MODE)`

Макроопределение:

```
((QUAL_MODE) == GPIO_QualMode_3sample) || \
((QUAL_MODE) == GPIO_QualMode_6sample))
```

Макрос проверки аргументов типа `GPIO_QualMode_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 258

Используется в `GPIO_QualConfig()`.

8.14.2.36 `#define IS_GPIO_SYNC(SYNC)`

Макроопределение:

```
((SYNC) == GPIO_Sync_Dis) || \
((SYNC) == GPIO_Sync_En))
```

Макрос проверки аргументов типа `GPIO_Sync_TypeDef`.

См. определение в файле niietcm4_gpio.h строка 224

8.14.3 Перечисления

8.14.3.1 `enum BitAction`

Тип, определяющий состояния бита.

Элементы перечислений

`Bit_CLEAR` Бит очищен.

`Bit_SET` Бит установлен.

См. определение в файле niietcm4_gpio.h строка 62

8.14.3.2 `enum GPIO_AltFunc_TypeDef`

Выбор номера альтернативной функции пина.

Элементы перечислений

`GPIO_AltFunc_1` Альтернативная функция 1.

`GPIO_AltFunc_2` Альтернативная функция 2.

`GPIO_AltFunc_3` Альтернативная функция 3.

См. определение в файле niietcm4_gpio.h строка 265

8.14.3.3 `enum GPIO_Dir_TypeDef`

Выбор направления работы пина.

Элементы перечислений

`GPIO_Dir_In` Пин настроен на вход.

`GPIO_Dir_Out` Пин настроен на выход.

См. определение в файле niietcm4_gpio.h строка 78

8.14.3.4 enum GPIO_IntPol_TypeDef

Выбор полярности события для возникновения прерывания.

Элементы перечислений

GPIO_IntPol_Neg Прерывание по низкому уровню или отрицательному фронту.

GPIO_IntPol_Pos Прерывание по высокому уровню или положительному фронту.

См. определение в файле niitcm4_gpio.h строка 129

8.14.3.5 enum GPIO_IntType_TypeDef

Выбор события для возникновения прерывания.

Элементы перечислений

GPIO_IntType_Level Прерывание по уровню.

GPIO_IntType_Edge Прерывание по перепаду.

См. определение в файле niitcm4_gpio.h строка 112

8.14.3.6 enum GPIO_Load_TypeDef

Выбор максимальной нагрузочной способности пина.

Элементы перечислений

GPIO_Load_8mA Максимальный ток 8mA.

GPIO_Load_16mA Максимальный ток 16mA.

См. определение в файле niitcm4_gpio.h строка 163

8.14.3.7 enum GPIO_Mode_TypeDef

Выбор режима работы пина.

Элементы перечислений

GPIO_Mode_IO Пин в режиме ввода-вывода.

GPIO_Mode_AltFunc Пин в режиме альтернативной функции.

См. определение в файле niitcm4_gpio.h строка 95

8.14.3.8 enum GPIO_Out_TypeDef

Включение выхода пина.

Элементы перечислений

GPIO_Out_Dis Пин в третьем состоянии.

GPIO_Out_En Пин работает как выход.

См. определение в файле niitcm4_gpio.h строка 146

8.14.3.9 enum GPIO_OutMode_TypeDef

Выбор режима работы выходных каскадов.

Элементы перечислений

GPIO_OutMode_PP Режим пуш-пулл.

GPIO_OutMode_OD Режим открытого коллектора.

См. определение в файле niietcm4_gpio.h строка 180

8.14.3.10 enum GPIO_PullUp_TypeDef

Включение подтяжки к питанию.

Элементы перечислений

GPIO_PullUp_Dis Внутренняя подтяжка к питанию выключена.

GPIO_PullUp_En Внутренняя подтяжка к питанию включена.

См. определение в файле niietcm4_gpio.h строка 197

8.14.3.11 enum GPIO_Qual_TypeDef

Включение входного фильтра.

Элементы перечислений

GPIO_Qual_Dis Входной фильтр выключен.

GPIO_Qual_En Входной фильтр включен.

См. определение в файле niietcm4_gpio.h строка 231

8.14.3.12 enum GPIO_QualMode_TypeDef

Выбор режима работы входного фильтра.

Элементы перечислений

GPIO_QualMode_3sample Используется 3 отсчета для фильтрации.

GPIO_QualMode_6sample Используется 6 отсчетов для фильтрации.

См. определение в файле niietcm4_gpio.h строка 248

8.14.3.13 enum GPIO_Sync_TypeDef

Включение режима пересинхронизации входов через 2 триггера-защелки.

Элементы перечислений

GPIO_Sync_Dis Пересинхронизация входа выключена.

GPIO_Sync_En Пересинхронизация входа включена.

См. определение в файле niietcm4_gpio.h строка 214

8.14.4 Функции

8.14.4.1 `void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 626

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.14.4.2 void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)

Устанавливает все регистры выбранного GPIOx значениями по умолчанию.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 123

Перекрестные ссылки GPIO_DATAOUT_Reset_Value, GPIO_GPIODEN0_Reset_Value, GPIO_GPIODEN1_Reset_Value, GPIO_GPIODEN2_Reset_Value, GPIO_GPIODEN3_Reset_Value, GPIO_GPIOODCTLx_Reset_Value, GPIO_GPIOODSCTLx_Reset_Value, GPIO_GPIOPCTLx_Reset_Value, GPIO_GPIOPUCTLx_Reset_Value, GPIO_GPIOQEx_Reset_Value, GPIO_GPIOQMx_Reset_Value, GPIO_GPIOQPx_Reset_Value, GPIO_GPIOSEx_Reset_Value, GPIO_Regs_A_C_E_G_Mask, GPIO_Regs_B_D_F_H_Mask и IS_GPIO_ALL_PERIPH.

8.14.4.3 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 331

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_In, GPIO_Dir_Out, GPIO_Init_TypeDef::GPIO_Load, GPIO_Load_16mA, GPIO_Load_8mA, GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Mode_IO, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_Dis, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_OutMode, GPIO_OutMode_OD, GPIO_OutMode_PP, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Init_TypeDef::GPIO_PullUp, GPIO_PullUp_Dis, GPIO_PullUp_En, IS_GPIO_ALL_PERIPH, IS_GPIO_ALT_FUNC, IS_GPIO_DIR, IS_GPIO_LOAD, IS_GPIO_MODE, IS_GPIO_OUT, IS_GPIO_OUT_MODE, IS_GPIO_PIN и IS_GPIO_PULLUP.

Используется в RCC_SysClkDiv2Out().

```
8.14.4.4 void GPIO_ITCmd ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,  
    FunctionalState State )
```

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 913

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.14.4.5 `void GPIO_ITConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)`

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 876

Перекрестные ссылки `GPIO_IntPol_Neg`, `GPIO_IntPol_Pos`, `GPIO_IntType_Edge`, `GPIO_IntType_Level`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_INT_POL`, `IS_GPIO_INT_TYPE` и `IS_GPIO_PIN`.

8.14.4.6 `void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 938

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.14.4.7 `void GPIO_QualCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)`

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 753

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.14.4.8 `void GPIO_QualConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)`

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 664

Перекрестные ссылки `GPIO_QualMode_3sample`, `GPIO_QualMode_6sample`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_PIN`, `IS_GPIO_QUAL_MODE` и `IS_GPIO_QUAL_PERIOD`.

8.14.4.9 `uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)`

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx

См. определение в файле `niietcm4_gpio.c` строка 503

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

8.14.4.10 `uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне А..Н.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина	
---------------------------	--

См. определение в файле niietcm4_gpio.c строка 477

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN и IS_GPIO_ALL_PERIPH.

8.14.4.11 uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)

Чтение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне А..Н.
MaskVal	Значение маски чтения.

Возвращаемые значения

Состояние порта GPIOx с учетом маски	
--------------------------------------	--

См. определение в файле niietcm4_gpio.c строка 518

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.14.4.12 void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне А..Н.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 609

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.14.4.13 void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)

Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 456

Перекрестные ссылки `GPIO_InitTypeDef::GPIO_AltFunc`, `GPIO_AltFunc_1`, `GPIO_InitTypeDef::GPIO_Dir`, `GPIO_Dir_In`, `GPIO_InitTypeDef::GPIO_Load`, `GPIO_Load_8mA`, `GPIO_InitTypeDef::GPIO_Mode`, `GPIO_Mode_IO`, `GPIO_InitTypeDef::GPIO_Out`, `GPIO_Out_Dis`, `GPIO_InitTypeDef::GPIO_OutMode`, `GPIO_OutMode_PP`, `GPIO_InitTypeDef::GPIO_Pin`, `GPIO_Pin_All`, `GPIO_InitTypeDef::GPIO_PullUp` и `GPIO_PullUp_Dis`.

Используется в `RCC_SysClkDiv2Out()`.

8.14.4.14 `void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)`

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

<code>GPIOx</code>	выбор порта, где x лежит в диапазоне A..H.
<code>GPIO_Pin</code>	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
<code>State</code>	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 814

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.14.4.15 `void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Переключение выбранных пинов в противоположное состояние.

Аргументы

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне A..H.
<code>GPIO_Pin</code>	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 643

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.14.4.16 `void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)`

Изменение состояния выбранного порта `GPIOx`.

Аргументы

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне A..H.
<code>PortVal</code>	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.14.4.17 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 546

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN, IS_GPIO_ALL_PERIPH и IS_GPIO_BIT_ACTION.

8.14.4.18 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 586

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.15 Файл niietcm4_rcc.c

Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера.

```
#include "niietcm4_rcc.h"
```

Макросы

- `#define RCC_PLL_CTRL_Reset_Value ((uint32_t)0x00000000)`
- `#define RCC_PLL_OD_Reset_Value ((uint32_t)0x00000000)`

- `#define RCC_PLL_NR_Reset_Value ((uint32_t)0x00000000)`
- `#define RCC_PLL_NF_Reset_Value ((uint32_t)0x00000000)`

Функции

- `uint32_t RCC_WaitClkChange (RCC_SysClk_TypeDef RCC_SysClk)`
Процедура ожидания смены источника тактового сигнала
- `void RCC_SysClkDiv2Out (FunctionalState State)`
Включение генерации тактового сигнала с частотой равной половине системной на выводе H[0].
Функция использует драйвер GPIO для настройки выхода.
- `OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)`
Автоматическая конфигурация PLL для получения желаемой системной частоты.
- `void RCC_PLLInit (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)`
Инициализирует PLL согласно параметрам структуры `RCC_PLLInit_Struct`.
- `void RCC_PLLStructInit (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)`
Заполнение каждого члена структуры `RCC_PLLInit_Struct` значениями по умолчанию.
- `void RCC_PLLDeInit ()`
Устанавливает все регистры PLL значениями по умолчанию.
- `void RCC_PLLPowerDownCmd (FunctionalState State)`
Управление режимом PowerDown PLL.
- `void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)`
Включение тактирования выбранного блока периферии.
- `OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)`
Выбор источника для системного тактового сигнала.
- `RCC_SysClk_TypeDef RCC_SysClkStatus ()`
Текущий источник системного тактового сигнала.
- `void RCC_USBClkConfig (RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)`
Настройка источника тактового сигнала для USB.
- `void RCC_USBClkCmd (FunctionalState State)`
Включение тактирования USB.
- `void RCC_UARTClkSel (NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)`
Настройка источника тактового сигнала для выбранного UART.
- `void RCC_UARTClkDivConfig (NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)`
Настройка делителя тактового сигнала для выбранного UART.
- `void RCC_UARTClkCmd (NT_UART_TypeDef *UARTx, FunctionalState State)`
Включение тактирования UART.
- `void RCC_SPIClkSel (NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)`
Настройка источника тактового сигнала для выбранного SPI.
- `void RCC_SPIClkDivConfig (NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)`
Настройка делителя тактового сигнала для выбранного SPI.
- `void RCC_SPIClkCmd (NT_SPI_TypeDef *SPIx, FunctionalState State)`
Включение тактирования SPI.
- `void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)`
Настройка делителя тактового сигнала для выбранного ADC.
- `void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)`
Включение тактирования ADC.
- `void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)`
Вывод из состояния сброса периферийных блоков.

8.15.1 Подробное описание

Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

06.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.15.2 Макросы

8.15.2.1 `#define RCC_PLL_CTRL_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_CTRL

См. определение в файле niietcm4_rcc.c строка 54

Используется в RCC_PLLDeInit().

8.15.2.2 `#define RCC_PLL_NF_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_NF

См. определение в файле niietcm4_rcc.c строка 57

Используется в RCC_PLLDeInit().

8.15.2.3 `#define RCC_PLL_NR_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_NR

См. определение в файле niietcm4_rcc.c строка 56

Используется в RCC_PLLDeInit().

8.15.2.4 `#define RCC_PLL_OD_Reset_Value ((uint32_t)0x00000000)`

Значение по сбросу регистра PLL_OD

См. определение в файле `niietcm4_rcc.c` строка 55

Используется в `RCC_PLLDeInit()`.

8.15.3 Функции

8.15.3.1 `void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)`

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 779

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_ADC_CLK`, `RCC_ADCClk_0`, `RCC_ADCClk_1`, `RCC_ADCClk_10`, `RCC_ADCClk_2`, `RCC_ADCClk_3`, `RCC_ADCClk_4`, `RCC_ADCClk_5`, `RCC_ADCClk_6`, `RCC_ADCClk_7`, `RCC_ADCClk_8` и `RCC_ADCClk_9`.

8.15.3.2 `void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 695

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_ADC_CLK`, `IS_RCC_CLK_DIV`, `RCC_ADCClk_0`, `RCC_ADCClk_1`, `RCC_ADCClk_10`, `RCC_ADCClk_2`, `RCC_ADCClk_3`, `RCC_ADCClk_4`, `RCC_ADCClk_5`, `RCC_ADCClk_6`, `RCC_ADCClk_7`, `RCC_ADCClk_8` и `RCC_ADCClk_9`.

8.15.3.3 `void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)`

Включение тактирования выбранного блока периферии.

Внимание

Блоки UART , SPI, ADC, USB управляются отдельно.

- [Тактирование UART](#)
- [Тактирование SPI](#)
- [Тактирование ADC](#)
- [Тактирование USB](#)

Аргументы

RCC_Periph↔ Clk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_↔ TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 342

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_RCC_PERIPH_CLK.

8.15.3.4 void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_Periph↔ Rst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_↔ PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 869

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_PERIPH_RST и RCC_PeriphRst_↔
ETH.

Используется в CAP_DeInit() и UART_DeInit().

8.15.3.5 OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)

Автоматическая конфигурация PLL для получения желаемой системной частоты.

С учетом данных об источнике частоты для PLL, а также о значении желаемой частоты, вычисляются все необходимые коэффициенты.

Внимание

Если Freq < 50 МГц, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

RCC_PLLRef	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из RCC_PLLRef_TypeDef .
SysFreq	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 142

Перекрестные ссылки EXT_OSC_VALUE, IS_RCC_PLL_REF, IS_RCC_SYS_FREQ, RCC_CLK_PLL_STABLE_TIMEOUT, RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit(), RCC_PLLInit_TypeDef::RCC_PLLNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLNO_Div2, RCC_PLLNO_Div4, RCC_PLLInit_TypeDef::RCC_PLLNR, RCC_PLLInit_TypeDef::RCC_PLLRef, RCC_PLLRef_ETH_25MHz, RCC_PLLRef_USB_60MHz, RCC_PLLRef_USB_CLK, RCC_PLLRef_XI_OSC, RCC_SysClk_PLL, RCC_SysClk_PLLDIV и RCC_SysClkSel().

8.15.3.6 void RCC_PLLDeInit ()

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 298

Перекрестные ссылки RCC_PLL_CTRL_Reset_Value, RCC_PLL_NF_Reset_Value, RCC_PLL_NR_Reset_Value и RCC_PLL_OD_Reset_Value.

8.15.3.7 void RCC_PLLInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Инициализирует PLL согласно параметрам структуры RCC_PLLInit_Struct.

Значение выходной частоты PLL вычисляется с использованием значений опорного NR и выходного NO делителей, а также делителя обратной связи NF по формуле:

$$F_{OUT} = (F_{IN} \times NF) / (NO \times NR),$$

где F_{IN} – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- $3,2 \text{ МГц} < F_{IN} < 150 \text{ МГц}$,
- $800 \text{ КГц} < F_{REF} < 8 \text{ МГц}$,
- $200 \text{ МГц} < F_{VCO} < 500 \text{ МГц}$,

где частота фазового детектора F_{REF} вычисляется по формуле:

$$\text{FREF} = \text{FIN} / (2 \times \text{NR}),$$

а частота FVCO вычисляется по формуле:

$$\text{FVCO} = \text{FIN} \times (\text{NF} / \text{NR})$$

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 262

Перекрестные ссылки IS_RCC_PLL_NF, IS_RCC_PLL_NO, IS_RCC_PLL_NR, IS_RCC_PL↔L_REF, IS_RCC_PLLDIV, RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit_TypeDef::RC↔C_PLLNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLInit_TypeDef::RCC_PLLNR и RC↔C_PLLInit_TypeDef::RCC_PLLRef.

Используется в RCC_PLLAutoConfig().

8.15.3.8 void RCC_PLLPowerDownCmd (FunctionalState State)

Управление режимом PowerDown PLL.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 313

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.15.3.9 void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 284

Перекрестные ссылки RCC_PLLInit_TypeDef::RCC_PLLDiv, RCC_PLLInit_TypeDef::RCC_PL↔LNF, RCC_PLLInit_TypeDef::RCC_PLLNO, RCC_PLLNO_Disable, RCC_PLLInit_TypeDef::R↔CC_PLLNR, RCC_PLLInit_TypeDef::RCC_PLLRef и RCC_PLLRef_XI_OSC.

8.15.3.10 void RCC_SPIClkCmd (NT_SPI_TypeDef * SPIx, FunctionalState State)

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 648

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_SPI_ALL_PERIPH.

```
8.15.3.11 void RCC_SPIClkDivConfig ( NT_SPI_TypeDef * SPIx, uint32_t DivVal,
FunctionalState DivState )
```

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 610

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_SPI_ALL_PERIPH.

```
8.15.3.12 void RCC_SPIClkSel ( NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk
)
```

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 569

Перекрестные ссылки IS_RCC_SPI_CLK и IS_SPI_ALL_PERIPH.

```
8.15.3.13 void RCC_SysClkDiv2Out ( FunctionalState State )
```

Включение генерации тактового сигнала с частотой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState: <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 106

Перекрестные ссылки GPIO_InitTypeDef::GPIO_AltFunc, GPIO_AltFunc_2, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_Out, GPIO_Init(), GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_0, GPIO_StructInit() и IS_FUNCTIONAL_STATE.

8.15.3.14 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_TypeDef .
------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 365

Перекрестные ссылки IS_RCC_SYS_CLK и RCC_WaitClkChange().

Используется в RCC_PLLAutoConfig().

8.15.3.15 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле niietcm4_rcc.c строка 394

8.15.3.16 void RCC_UARTClkCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 526

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.15.3.17 void RCC_UARTClkDivConfig (NT_UART_TypeDef * UARTx, uint32_t DivVal, FunctionalState DivState)

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 488

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_UART_ALL_PERIPH.

8.15.3.18 void RCC_UARTClkSel (NT_UART_TypeDef * UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)

Настройка источника тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
RCC_UARTClk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 448

Перекрестные ссылки IS_RCC_UART_CLK и IS_UART_ALL_PERIPH.

8.15.3.19 void RCC_USBClkCmd (FunctionalState State)

Включение тактирования USB.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 425

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.15.3.20 void RCC_USBClkConfig (RCC_USBClk_TypeDef RCC_USBClk,
RCC_USBFreq_TypeDef RCC_USBFreq)

Настройка источника тактового сигнала для USB.

Аргументы

RCC_USBClk	Выбор источника тактирования. Параметр принимает любое значение из RCC_USBClk_TypeDef .
RCC_USB\leftrightarrowFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 408

Перекрестные ссылки IS_RCC_USB_CLK и IS_RCC_USB_FREQ.

8.15.3.21 uint32_t RCC_WaitClkChange (RCC_SysClk_TypeDef RCC_SysClk)

Процедура ожидания смены источника тактового сигнала

Возвращаемые значения

timeout	Значение остатка времени после ожидания.
---------	--

См. определение в файле niietcm4_rcc.c строка 77

Перекрестные ссылки RCC_CLK_CHANGE_TIMEOUT.

Используется в RCC_SysClkSel().

8.16 Файл niietcm4_rcc.h

Файл содержит все прототипы функций для RCC (Reset & Clock Control).

```
#include "niietcm4.h"
```

Структуры данных

- struct [RCC_PLLInit_TypeDef](#)
Структура инициализации PLL.

Макросы

- #define [RCC_CLK_CHANGE_TIMEOUT](#) ((uint32_t)10000)
- #define [RCC_CLK_PLL_STABLE_TIMEOUT](#) ((uint32_t)100)
- #define [IS_RCC_PLL_REF](#)(PLL_REF)
Макрос проверки аргументов типа [RCC_PLLRef_TypeDef](#).
- #define [IS_RCC_PLL_NO](#)(PLL_NO)

- Макрос проверки аргументов типа `RCC_PLLNO_TypeDef`.
- `#define IS_RCC_UART_CLK(UART_CLK)`
Макрос проверки аргументов типа `RCC_UARTClk_TypeDef`.
- `#define IS_RCC_SPI_CLK(SPI_CLK)`
Макрос проверки аргументов типа `RCC_SPIClk_TypeDef`.
- `#define IS_RCC_USB_CLK(USB_CLK)`
Макрос проверки аргументов типа `RCC_USBClk_TypeDef`.
- `#define IS_RCC_USB_FREQ(USB_FREQ)`
Макрос проверки аргументов типа `RCC_USBFreq_TypeDef`.
- `#define IS_RCC_ADC_CLK(ADC_CLK)`
Макрос проверки аргументов типа `RCC_ADCClk_TypeDef`.
- `#define IS_RCC_SYS_CLK(SYS_CLK)`
Макрос проверки аргументов типа `RCC_SysClk_TypeDef`.
- `#define IS_RCC_PERIPH_CLK(PERIPH_CLK)`
Макрос проверки аргументов типа `RCC_PeriphClk_TypeDef`.
- `#define IS_RCC_PERIPH_RST(PERIPH_RST)`
Макрос проверки аргументов типа `RCC_PeriphRst_TypeDef`.
- `#define IS_RCC_PLLDIV(PLLDIV) (((PLLDIV) & ((uint32_t)0xFFFFF00)) == ((uint32_t)0x00))`
Макрос проверки значения выходного делителя PLL на попадание в допустимый диапазон.
- `#define IS_RCC_PLL_NR(PLL_NR) (((PLL_NR) <= ((uint32_t)33)) && ((PLL_NR) >= ((uint32_t)2)))`
Макрос проверки значения опорного делителя PLL на попадание в допустимый диапазон.
- `#define IS_RCC_PLL_NF(PLL_NF) (((PLL_NF) <= ((uint32_t)513)) && ((PLL_NF) >= ((uint32_t)2)))`
Макрос проверки значения делителя OC PLL на попадание в допустимый диапазон.
- `#define IS_RCC_CLK_DIV(CLK_DIV) ((CLK_DIV) < ((uint32_t)64))`
Макрос проверки значения делителя тактового сигнала на попадание в допустимый диапазон.
- `#define IS_RCC_SYS_FREQ(SYS_FREQ) (((SYS_FREQ) < ((uint32_t)20000000)) && ((SYS_FREQ) >= ((uint32_t)1000000)))`
Макрос проверки значения желаемой частоты при автонастройке в допустимый диапазон.

Перечисления

- `enum RCC_PLLRef_TypeDef { RCC_PLLRef_XI_OSC, RCC_PLLRef_USB_CLK, RCC_PLLRef_USB_60MHz, RCC_PLLRef_ETH_25MHz }`
Выбор источника опорного сигнала PLL.
- `enum RCC_PLLNO_TypeDef { RCC_PLLNO_Disable, RCC_PLLNO_Div2, RCC_PLLNO_Div4 = 3 }`
Выходной делитель NO.
- `enum RCC_UARTClk_TypeDef { RCC_UARTClk_SYSCLK, RCC_UARTClk_XI_OSC, RCC_UARTClk_USB_CLK, RCC_UARTClk_USB_60MHz }`
Выбор источника тактирования для UART.
- `enum RCC_SPIClk_TypeDef { RCC_SPIClk_SYSCLK, RCC_SPIClk_XI_OSC, RCC_SPIClk_USB_CLK, RCC_SPIClk_USB_60MHz }`
Выбор источника тактирования для SPI.
- `enum RCC_USBClk_TypeDef { RCC_USBClk_XI_OSC, RCC_USBClk_USB_CLK }`
Выбор источника тактирования для USB.
- `enum RCC_USBFreq_TypeDef { RCC_USBFreq_12MHz, RCC_USBFreq_24MHz }`
Выбор фиксированной частоты на входе CLK_USB.

- enum `RCC_ADCClk_TypeDef` {
`RCC_ADCClk_0`, `RCC_ADCClk_1`, `RCC_ADCClk_2`, `RCC_ADCClk_3`,
`RCC_ADCClk_4`, `RCC_ADCClk_5`, `RCC_ADCClk_6`, `RCC_ADCClk_7`,
`RCC_ADCClk_8`, `RCC_ADCClk_9`, `RCC_ADCClk_10`, `RCC_ADCClk_11` }

Выбор модуля ADC для настройки его тактового сигнала.

- enum `RCC_SysClk_TypeDef` {
`RCC_SysClk_CPE_Sel`, `RCC_SysClk_POR`, `RCC_SysClk_XI_OSC`, `RCC_SysClk_PLL`,
`RCC_SysClk_PLLDIV`, `RCC_SysClk_USB60MHz`, `RCC_SysClk_USB_CLK`, `RCC_SysClk_↵`
`_ETH25MHz` }

Выбор источника системной частоты.

- enum `RCC_PeriphClk_TypeDef` {
`RCC_PeriphClk_QEP0` = ((uint32_t)(1<<1)), `RCC_PeriphClk_QEP1` = ((uint32_t)(1<<2)),
`RCC_PeriphClk_CMP` = ((uint32_t)(1<<9)), `RCC_PeriphClk_PWM0` = ((uint32_t↵
t)(1<<10)),
`RCC_PeriphClk_PWM1` = ((uint32_t)(1<<11)), `RCC_PeriphClk_PWM2` = ((uint32_t↵
t)(1<<12)), `RCC_PeriphClk_PWM3` = ((uint32_t)(1<<13)), `RCC_PeriphClk_PWM4` =
((uint32_t)(1<<14)),
`RCC_PeriphClk_PWM5` = ((uint32_t)(1<<15)), `RCC_PeriphClk_PWM6` = ((uint32_t↵
t)(1<<16)), `RCC_PeriphClk_PWM7` = ((uint32_t)(1<<17)), `RCC_PeriphClk_PWM8` =
((uint32_t)(1<<18)),
`RCC_PeriphClk_WD` = ((uint32_t)(1<<19)), `RCC_PeriphClk_I2C0` = ((uint32_t)(1<<20)),
`RCC_PeriphClk_I2C1` = ((uint32_t)(1<<21)), `RCC_PeriphClk_ADC` = ((uint32_t)(1<<24))
}

Управление тактированием периферийных блоков

- enum `RCC_PeriphRst_TypeDef` {
`RCC_PeriphRst_WD`, `RCC_PeriphRst_I2C0`, `RCC_PeriphRst_I2C1`, `RCC_PeriphRst_USB`,
`RCC_PeriphRst_Timer0`, `RCC_PeriphRst_Timer1`, `RCC_PeriphRst_Timer2`, `RCC_Periph↵`
`Rst_UART0`,
`RCC_PeriphRst_UART1`, `RCC_PeriphRst_UART2`, `RCC_PeriphRst_UART3`, `RCC_Periph↵`
`Rst_SPI0`,
`RCC_PeriphRst_SPI1`, `RCC_PeriphRst_SPI2`, `RCC_PeriphRst_SPI3`, `RCC_PeriphRst_ETH`,
`RCC_PeriphRst_QEP0`, `RCC_PeriphRst_QEP1`, `RCC_PeriphRst_PWM0`, `RCC_PeriphRst↵`
`_PWM1`,
`RCC_PeriphRst_PWM2`, `RCC_PeriphRst_PWM3`, `RCC_PeriphRst_PWM4`, `RCC_Periph↵`
`Rst_PWM5`,
`RCC_PeriphRst_PWM6`, `RCC_PeriphRst_PWM7`, `RCC_PeriphRst_PWM8`, `RCC_Periph↵`
`Rst_CAP0`,
`RCC_PeriphRst_CAP1`, `RCC_PeriphRst_CAP2`, `RCC_PeriphRst_CAP3`, `RCC_PeriphRst_↵`
`CAP4`,
`RCC_PeriphRst_CAP5`, `RCC_PeriphRst_CMP` }

Управление сбросом периферийных блоков

Функции

- void `RCC_SysClkDiv2Out` (FunctionalState State)
Включение генерации тактового сигнала с частотой равной половине системной на выводе H[0].
Функция использует драйвер GPIO для настройки выхода.
- OperationStatus `RCC_PLLAutoConfig` (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t Sys↵
Freq)
Автоматическая конфигурация PLL для получения желаемой системной частоты.
- void `RCC_PLLInit` (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)
Инициализирует PLL согласно параметрам структуры RCC_PLLInit_Struct.
- void `RCC_PLLDeInit` ()
Устанавливает все регистры PLL значениями по умолчанию.
- void `RCC_PLLStructInit` (RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)

- Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.
- void [RCC_PLLPowerDownCmd](#) ([FunctionalState](#) State)
Управление режимом PowerDown PLL.
- void [RCC_PeriphClkCmd](#) ([RCC_PeriphClk_TypeDef](#) RCC_PeriphClk, [FunctionalState](#) State)
Включение тактирования выбранного блока периферии.
- [OperationStatus](#) [RCC_SysClkSel](#) ([RCC_SysClk_TypeDef](#) RCC_SysClk)
Выбор источника для системного тактового сигнала.
- [RCC_SysClk_TypeDef](#) [RCC_SysClkStatus](#) ()
Текущий источник системного тактового сигнала.
- void [RCC_USBClkConfig](#) ([RCC_USBClk_TypeDef](#) RCC_USBClk, [RCC_USBFreq_TypeDef](#) RCC_USBFreq)
Настройка источника тактового сигнала для USB.
- void [RCC_USBClkCmd](#) ([FunctionalState](#) State)
Включение тактирования USB.
- void [RCC_UARTClkSel](#) ([NT_UART_TypeDef](#) *UARTx, [RCC_UARTClk_TypeDef](#) RCC_UARTClk)
Настройка источника тактового сигнала для выбранного UART.
- void [RCC_UARTClkDivConfig](#) ([NT_UART_TypeDef](#) *UARTx, uint32_t DivVal, [FunctionalState](#) DivState)
Настройка делителя тактового сигнала для выбранного UART.
- void [RCC_UARTClkCmd](#) ([NT_UART_TypeDef](#) *UARTx, [FunctionalState](#) State)
Включение тактирования UART.
- void [RCC_SPIClkSel](#) ([NT_SPI_TypeDef](#) *SPIx, [RCC_SPIClk_TypeDef](#) RCC_SPIClk)
Настройка источника тактового сигнала для выбранного SPI.
- void [RCC_SPIClkDivConfig](#) ([NT_SPI_TypeDef](#) *SPIx, uint32_t DivVal, [FunctionalState](#) DivState)
Настройка делителя тактового сигнала для выбранного SPI.
- void [RCC_SPIClkCmd](#) ([NT_SPI_TypeDef](#) *SPIx, [FunctionalState](#) State)
Включение тактирования SPI.
- void [RCC_ADCClkDivConfig](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, uint32_t DivVal, [FunctionalState](#) DivState)
Настройка делителя тактового сигнала для выбранного ADC.
- void [RCC_ADCClkCmd](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, [FunctionalState](#) State)
Включение тактирования ADC.
- void [RCC_PeriphRstCmd](#) ([RCC_PeriphRst_TypeDef](#) RCC_PeriphRst, [FunctionalState](#) State)
Вывод из состояния сброса периферийных блоков.

8.16.1 Подробное описание

Файл содержит все прототипы функций для RCC (Reset & Clock Control).

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.16.2 Макросы

8.16.2.1 #define IS_RCC_ADC_CLK(ADC_CLK)

Макроопределение:

```
((ADC_CLK) == RCC_ADCClk_0) || \
((ADC_CLK) == RCC_ADCClk_1) || \
((ADC_CLK) == RCC_ADCClk_2) || \
((ADC_CLK) == RCC_ADCClk_3) || \
((ADC_CLK) == RCC_ADCClk_4) || \
((ADC_CLK) == RCC_ADCClk_5) || \
((ADC_CLK) == RCC_ADCClk_6) || \
((ADC_CLK) == RCC_ADCClk_7) || \
((ADC_CLK) == RCC_ADCClk_8) || \
((ADC_CLK) == RCC_ADCClk_9) || \
((ADC_CLK) == RCC_ADCClk_10) || \
((ADC_CLK) == RCC_ADCClk_11))
```

Макрос проверки аргументов типа `RCC_ADCClk_TypeDef`.

См. определение в файле `niietcm4_rcc.h` строка 204

Используется в `RCC_ADCClkCmd()` и `RCC_ADCClkDivConfig()`.

8.16.2.2 #define IS_RCC_PERIPH_CLK(PERIPH_CLK)

Макроопределение:

```
((PERIPH_CLK) == RCC_PeriphClk_QEP0) || \
((PERIPH_CLK) == RCC_PeriphClk_QEP1) || \
((PERIPH_CLK) == RCC_PeriphClk_CMP) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM0) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM1) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM2) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM4) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM5) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM6) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM7) || \
((PERIPH_CLK) == RCC_PeriphClk_PWM8) || \
((PERIPH_CLK) == RCC_PeriphClk_WD) || \
((PERIPH_CLK) == RCC_PeriphClk_I2C0) || \
((PERIPH_CLK) == RCC_PeriphClk_I2C1) || \
((PERIPH_CLK) == RCC_PeriphClk_ADC))
```

Макрос проверки аргументов типа `RCC_PeriphClk_TypeDef`.

См. определение в файле niietcm4_rcc.h строка 274

Используется в RCC_PeriphClkCmd().

8.16.2.3 #define IS_RCC_PLL_NO(PLL_NO)

Макроопределение:

```
((((PLL_NO) == RCC_PLLNO_Disable) || \
  ((PLL_NO) == RCC_PLLNO_Div2) || \
  ((PLL_NO) == RCC_PLLNO_Div4))
```

Макрос проверки аргументов типа RCC_PLLNO_TypeDef.

См. определение в файле niietcm4_rcc.h строка 100

Используется в RCC_PLLInit().

8.16.2.4 #define IS_RCC_PLL_REF(PLL_REF)

Макроопределение:

```
((((PLL_REF) == RCC_PLLRef_XI_OSC) || \
  ((PLL_REF) == RCC_PLLRef_USB_CLK) || \
  ((PLL_REF) == RCC_PLLRef_USB_60MHz) || \
  ((PLL_REF) == RCC_PLLRef_ETH_25MHz))
```

Макрос проверки аргументов типа RCC_PLLRef_TypeDef.

См. определение в файле niietcm4_rcc.h строка 79

Используется в RCC_PLLAutoConfig() и RCC_PLLInit().

8.16.2.5 #define IS_RCC_SPI_CLK(SPI_CLK)

Макроопределение:

```
((((SPI_CLK) == RCC_SPIClk_SYSCLK) || \
  ((SPI_CLK) == RCC_SPIClk_XI_OSC) || \
  ((SPI_CLK) == RCC_SPIClk_USB_CLK) || \
  ((SPI_CLK) == RCC_SPIClk_USB_60MHz))
```

Макрос проверки аргументов типа RCC_SPIClk_TypeDef.

См. определение в файле niietcm4_rcc.h строка 141

Используется в RCC_SPIClkSel().

8.16.2.6 #define IS_RCC_SYS_CLK(SYS_CLK)

Макроопределение:

```
((((SYS_CLK) == RCC_SysClk_CPE_Sel) || \
  ((SYS_CLK) == RCC_SysClk_POR) || \
  ((SYS_CLK) == RCC_SysClk_XI_OSC) || \
  ((SYS_CLK) == RCC_SysClk_PLL) || \
  ((SYS_CLK) == RCC_SysClk_PLLDIV) || \
  ((SYS_CLK) == RCC_SysClk_USB60MHz) || \
  ((SYS_CLK) == RCC_SysClk_USB_CLK) || \
  ((SYS_CLK) == RCC_SysClk_ETH25MHz))
```

Макрос проверки аргументов типа RCC_SysClk_TypeDef.

См. определение в файле niietcm4_rcc.h строка 237

Используется в RCC_SysClkSel().

8.16.2.7 #define IS_RCC_UART_CLK(UART_CLK)

Макроопределение:

```
((UART_CLK) == RCC_UARTClk_SYSClk) || \
    ((UART_CLK) == RCC_UARTClk_XI_OSC) || \
    ((UART_CLK) == RCC_UARTClk_USB_CLK) || \
    ((UART_CLK) == RCC_UARTClk_USB_60MHz))
```

Макрос проверки аргументов типа [RCC_UARTClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 120

Используется в `RCC_UARTClkSel()`.

8.16.2.8 #define IS_RCC_USB_CLK(USB_CLK)

Макроопределение:

```
((USB_CLK) == RCC_USBClk_XI_OSC) || \
    ((USB_CLK) == RCC_USBClk_USB_CLK))
```

Макрос проверки аргументов типа [RCC_USBClk_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 160

Используется в `RCC_USBClkConfig()`.

8.16.2.9 #define IS_RCC_USB_FREQ(USB_FREQ)

Макроопределение:

```
((USB_FREQ) == RCC_USBFreq_12MHz) || \
    ((USB_FREQ) == RCC_USBFreq_24MHz))
```

Макрос проверки аргументов типа [RCC_USBFreq_TypeDef](#).

См. определение в файле `niietcm4_rcc.h` строка 177

Используется в `RCC_USBClkConfig()`.

8.16.2.10 #define RCC_CLK_CHANGE_TIMEOUT ((uint32_t)10000)

Время ожидания смены источника тактирования

См. определение в файле `niietcm4_rcc.h` строка 52

Используется в `RCC_WaitClkChange()`.

8.16.2.11 #define RCC_CLK_PLL_STABLE_TIMEOUT ((uint32_t)100)

Время ожидания стабилизации выходной частоты PLL

См. определение в файле `niietcm4_rcc.h` строка 53

Используется в `RCC_PLLAutoConfig()`.

8.16.3 Перечисления

8.16.3.1 enum RCC_ADCClk_TypeDef

Выбор модуля ADC для настройки его тактового сигнала.

Элементы перечислений

RCC_ADCClk_0 Тактовый сигнал ADC 0
RCC_ADCClk_1 Тактовый сигнал ADC 1
RCC_ADCClk_2 Тактовый сигнал ADC 2
RCC_ADCClk_3 Тактовый сигнал ADC 3
RCC_ADCClk_4 Тактовый сигнал ADC 4
RCC_ADCClk_5 Тактовый сигнал ADC 5
RCC_ADCClk_6 Тактовый сигнал ADC 6
RCC_ADCClk_7 Тактовый сигнал ADC 7
RCC_ADCClk_8 Тактовый сигнал ADC 8
RCC_ADCClk_9 Тактовый сигнал ADC 9
RCC_ADCClk_10 Тактовый сигнал ADC 10
RCC_ADCClk_11 Тактовый сигнал ADC 11

См. определение в файле niietcm4_rcc.h строка 184

8.16.3.2 enum RCC_PeriphClk_TypeDef

Управление тактированием периферийных блоков

Элементы перечислений

RCC_PeriphClk_QEP0 Управление тактированием блока QEP 0
RCC_PeriphClk_QEP1 Управление тактированием блока QEP 1
RCC_PeriphClk_CMP Управление тактированием блока аналогового компаратора
RCC_PeriphClk_PWM0 Управление тактированием блока PWM 0
RCC_PeriphClk_PWM1 Управление тактированием блока PWM 1
RCC_PeriphClk_PWM2 Управление тактированием блока PWM 2
RCC_PeriphClk_PWM3 Управление тактированием блока PWM 3
RCC_PeriphClk_PWM4 Управление тактированием блока PWM 4
RCC_PeriphClk_PWM5 Управление тактированием блока PWM 5
RCC_PeriphClk_PWM6 Управление тактированием блока PWM 6
RCC_PeriphClk_PWM7 Управление тактированием блока PWM 7
RCC_PeriphClk_PWM8 Управление тактированием блока PWM 8
RCC_PeriphClk_WD Управление тактированием сторожевого таймера
RCC_PeriphClk_I2C0 Управление тактированием блока I2C 0
RCC_PeriphClk_I2C1 Управление тактированием блока I2C 1
RCC_PeriphClk_ADC Управление тактированием контроллера ADC

См. определение в файле niietcm4_rcc.h строка 250

8.16.3.3 enum RCC_PeriphRst_TypeDef

Управление сбросом периферийных блоков

Элементы перечислений

RCC_PeriphRst_WD Управление сбросом сторожевого таймера

RCC_PeriphRst_I2C0 Управление сбросом блока I2C 0
 RCC_PeriphRst_I2C1 Управление сбросом блока I2C 1
 RCC_PeriphRst_USB Управление сбросом блока USB
 RCC_PeriphRst_Timer0 Управление сбросом блока Timer 0
 RCC_PeriphRst_Timer1 Управление сбросом блока Timer 1
 RCC_PeriphRst_Timer2 Управление сбросом блока Timer 2
 RCC_PeriphRst_UART0 Управление сбросом блока UART 0
 RCC_PeriphRst_UART1 Управление сбросом блока UART 1
 RCC_PeriphRst_UART2 Управление сбросом блока UART 2
 RCC_PeriphRst_UART3 Управление сбросом блока UART 3
 RCC_PeriphRst_SPI0 Управление сбросом блока SPI 0
 RCC_PeriphRst_SPI1 Управление сбросом блока SPI 1
 RCC_PeriphRst_SPI2 Управление сбросом блока SPI 2
 RCC_PeriphRst_SPI3 Управление сбросом блока SPI 3
 RCC_PeriphRst_ETH Управление сбросом блока Ethernet
 RCC_PeriphRst_QEP0 Управление сбросом блока QEP 0
 RCC_PeriphRst_QEP1 Управление сбросом блока QEP 1
 RCC_PeriphRst_PWM0 Управление сбросом блока PWM 0
 RCC_PeriphRst_PWM1 Управление сбросом блока PWM 1
 RCC_PeriphRst_PWM2 Управление сбросом блока PWM 2
 RCC_PeriphRst_PWM3 Управление сбросом блока PWM 3
 RCC_PeriphRst_PWM4 Управление сбросом блока PWM 4
 RCC_PeriphRst_PWM5 Управление сбросом блока PWM 5
 RCC_PeriphRst_PWM6 Управление сбросом блока PWM 6
 RCC_PeriphRst_PWM7 Управление сбросом блока PWM 7
 RCC_PeriphRst_PWM8 Управление сбросом блока PWM 8
 RCC_PeriphRst_CAP0 Управление сбросом блока CAP 0
 RCC_PeriphRst_CAP1 Управление сбросом блока CAP 1
 RCC_PeriphRst_CAP2 Управление сбросом блока CAP 2
 RCC_PeriphRst_CAP3 Управление сбросом блока CAP 3
 RCC_PeriphRst_CAP4 Управление сбросом блока CAP 4
 RCC_PeriphRst_CAP5 Управление сбросом блока CAP 5
 RCC_PeriphRst_CMP Управление сбросом блока аналогового компаратора

См. определение в файле niietcm4_gcc.h строка 294

8.16.3.4 enum RCC_PLLNO_TypeDef

Выходной делитель NO.

Элементы перечислений

RCC_PLLNO_Disable Делитель NO выключен
 RCC_PLLNO_Div2 Коэффициент деления NO равен 2
 RCC_PLLNO_Div4 Коэффициент деления NO равен 4

См. определение в файле niietcm4_gcc.h строка 89

8.16.3.5 enum RCC_PLLRef_TypeDef

Выбор источника опорного сигнала PLL.

Элементы перечислений

RCC_PLLRef_XI_OSC Сигнал со входа XI_OSC
RCC_PLLRef_USB_CLK Сигнал с входной альтернативной функции CLK_USB
RCC_PLLRef_USB_60MHz Сигнал на выходе блока USB
RCC_PLLRef_ETH_25MHz Входной тактовый сигнал блока Ethernet

См. определение в файле niietcm4_rcc.h строка 67

8.16.3.6 enum RCC_SPIClk_TypeDef

Выбор источника тактирования для SPI.

Элементы перечислений

RCC_SPIClk_SYSCLK Текущая системная частота
RCC_SPIClk_XI_OSC Сигнал со входа XI_OSC
RCC_SPIClk_USB_CLK Сигнал с входной альтернативной функции CLK_USB
RCC_SPIClk_USB_60MHz Сигнал на выходе блока USB

См. определение в файле niietcm4_rcc.h строка 129

8.16.3.7 enum RCC_SysClk_TypeDef

Выбор источника системной частоты.

Элементы перечислений

RCC_SysClk_CPE_Sel Источник определяется состоянием вывода CPE: 0-POR, 1-XI_OSC
RCC_SysClk_POR Внутренний источник тактового сигнала
RCC_SysClk_XI_OSC Внешний источник тактового сигнала на входе XI_OSC
RCC_SysClk_PLL Выход блока PLL
RCC_SysClk_PLLDIV Выход блока PLL через делитель PLL DIV
RCC_SysClk_USB60MHz Выход блока USB 60 МГц
RCC_SysClk_USB_CLK Внешний источник тактового сигнала на входе CLK_USB
RCC_SysClk_ETH25MHz Входной тактовый сигнал блока Ethernet

См. определение в файле niietcm4_rcc.h строка 221

8.16.3.8 enum RCC_UARTClk_TypeDef

Выбор источника тактирования для UART.

Элементы перечислений

RCC_UARTClk_SYSCLK Текущая системная частота
RCC_UARTClk_XI_OSC Сигнал со входа XI_OSC
RCC_UARTClk_USB_CLK Сигнал с входной альтернативной функции CLK_USB
RCC_UARTClk_USB_60MHz Сигнал на выходе блока USB

См. определение в файле niietcm4_rcc.h строка 108

8.16.3.9 enum RCC_USBClk_TypeDef

Выбор источника тактирования для USB.

Элементы перечислений

RCC_USBClk_XI_OSC Сигнал со входа XI_OSC

RCC_USBClk_USB_CLK Сигнал с входной альтернативной функции CLK_USB

См. определение в файле niietcm4_rcc.h строка 150

8.16.3.10 enum RCC_USBFreq_TypeDef

Выбор фиксированной частоты на входе CLK_USB.

Элементы перечислений

RCC_USBFreq_12MHz 12 МГц сигнал на входе CLK_USB

RCC_USBFreq_24MHz 24 МГц сигнал на входе CLK_USB

См. определение в файле niietcm4_rcc.h строка 167

8.16.4 Функции

8.16.4.1 void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 779

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_ADC_CLK, RCC_ADCClk_0, RCC_ADCClk_1, RCC_ADCClk_10, RCC_ADCClk_2, RCC_ADCClk_3, RCC_ADCClk_4, RCC_ADCClk_5, RCC_ADCClk_6, RCC_ADCClk_7, RCC_ADCClk_8 и RCC_ADCClk_9.

8.16.4.2 void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
------------	---

DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 695

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_ADC_CLK, IS_RCC_CLK_DIV, RCC_ADCClk_0, RCC_ADCClk_1, RCC_ADCClk_10, RCC_ADCClk_2, RCC_ADCClk_3, RCC_ADCClk_4, RCC_ADCClk_5, RCC_ADCClk_6, RCC_ADCClk_7, RCC_ADCClk_8 и RCC_ADCClk_9.

8.16.4.3 void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)

Включение тактирования выбранного блока периферии.

Внимание

Блоки UART , SPI, ADC, USB управляются отдельно.

- [Тактирование UART](#)
- [Тактирование SPI](#)
- [Тактирование ADC](#)
- [Тактирование USB](#)

Аргументы

RCC_PeriphClk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 342

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_RCC_PERIPH_CLK.

8.16.4.4 void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_PeriphRst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 869

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_PERIPH_RST` и `RCC_PeriphRst_↔ETH`.

Используется в `CAP_DeInit()` и `UART_DeInit()`.

8.16.4.5 `OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)`

Автоматическая конфигурация PLL для получения желаемой системной частоты.

С учетом данных об источнике частоты для PLL, а также о значении желаемой частоты, вычисляются все необходимые коэффициенты.

Внимание

Если $\text{Freq} < 50$ МГц, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

<code>RCC_PLLRef</code>	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из RCC_PLLRef_TypeDef .
<code>SysFreq</code>	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 142

Перекрестные ссылки `EXT_OSC_VALUE`, `IS_RCC_PLL_REF`, `IS_RCC_SYS_FREQ`, `RCC_C↔LK_PLL_STABLE_TIMEOUT`, `RCC_PLLInit_TypeDef::RCC_PLLDiv`, `RCC_PLLInit()`, `RCC_↔PLLInit_TypeDef::RCC_PLLNF`, `RCC_PLLInit_TypeDef::RCC_PLLNO`, `RCC_PLLNO_Div2`, `R↔CC_PLLNO_Div4`, `RCC_PLLInit_TypeDef::RCC_PLLNR`, `RCC_PLLInit_TypeDef::RCC_PLL↔Ref`, `RCC_PLLRef_ETH_25MHz`, `RCC_PLLRef_USB_60MHz`, `RCC_PLLRef_USB_CLK`, `RCC↔_PLLRef_XI_OSC`, `RCC_SysClk_PLL`, `RCC_SysClk_PLLDIV` и `RCC_SysClkSel()`.

8.16.4.6 `void RCC_PLLEnInit ()`

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 298

Перекрестные ссылки `RCC_PLL_CTRL_Reset_Value`, `RCC_PLL_NF_Reset_Value`, `RCC_PLL↔_NR_Reset_Value` и `RCC_PLL_OD_Reset_Value`.

8.16.4.7 `void RCC_PLLInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)`

Инициализирует PLL согласно параметрам структуры `RCC_PLLInit_Struct`.

Значение выходной частоты PLL вычисляется с использованием значений опорного NR и выходного NO делителей, а также делителя обратной связи NF по формуле:

$$FOUT = (FIN \times NF) / (NO \times NR),$$

где FIN – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- $3,2 \text{ МГц} < FIN < 150 \text{ МГц}$,
- $800 \text{ КГц} < FREF < 8 \text{ МГц}$,
- $200 \text{ МГц} < FVCO < 500 \text{ МГц}$,

где частота фазового детектора $FREF$ вычисляется по формуле:

$$FREF = FIN / (2 \times NR),$$

а частота $FVCO$ вычисляется по формуле:

$$FVCO = FIN \times (NF / NR)$$

Аргументы

<code>RCC_PLLInit_Struct</code>	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
---------------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 262

Перекрестные ссылки `IS_RCC_PLL_NF`, `IS_RCC_PLL_NO`, `IS_RCC_PLL_NR`, `IS_RCC_PLL_REF`, `IS_RCC_PLLDIV`, `RCC_PLLInit_TypeDef::RCC_PLLDiv`, `RCC_PLLInit_TypeDef::RCC_PLLNF`, `RCC_PLLInit_TypeDef::RCC_PLLNO`, `RCC_PLLInit_TypeDef::RCC_PLLNR` и `RCC_PLLInit_TypeDef::RCC_PLLRef`.

Используется в `RCC_PLLAutoConfig()`.

8.16.4.8 `void RCC_PLLPowerDownCmd (FunctionalState State)`

Управление режимом PowerDown PLL.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 313

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

8.16.4.9 `void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)`

Заполнение каждого члена структуры `RCC_PLLInit_Struct` значениями по умолчанию.

Аргументы

<code>RCC_PLLInit_Struct</code>	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
---------------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 284

Перекрестные ссылки `RCC_PLLInit_TypeDef::RCC_PLLDiv`, `RCC_PLLInit_TypeDef::RCC_PL<LF`, `RCC_PLLInit_TypeDef::RCC_PLLNO`, `RCC_PLLNO_Disable`, `RCC_PLLInit_TypeDef::RCC_PLLNR`, `RCC_PLLInit_TypeDef::RCC_PLLRef` и `RCC_PLLRef_XI_OSC`.

8.16.4.10 `void RCC_SPIClkCmd (NT_SPI_TypeDef * SPIx, FunctionalState State)`

Включение тактирования SPI.

Аргументы

<code>SPIx</code>	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 648

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_SPI_ALL_PERIPH`.

8.16.4.11 `void RCC_SPIClkDivConfig (NT_SPI_TypeDef * SPIx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

<code>SPIx</code>	Выбор модуля SPI, где x лежит в диапазоне 0-3.
<code>DivVal</code>	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.

DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .
----------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 610

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_SPI_ALL_PERIPH.

8.16.4.12 void RCC_SPIClkSel (NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 569

Перекрестные ссылки IS_RCC_SPI_CLK и IS_SPI_ALL_PERIPH.

8.16.4.13 void RCC_SysClkDiv2Out (FunctionalState State)

Включение генерации тактового сигнала с частой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState: <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 106

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_AltFunc_2, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_Out, GPIO_Init(), GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_0, GPIO_StructInit() и IS_FUNCTIONAL_STATE.

8.16.4.14 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_Type ↵ Def .
------------	--

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 365

Перекрестные ссылки `IS_RCC_SYS_CLK` и `RCC_WaitClkChange()`.

Используется в `RCC_PLLAutoConfig()`.

8.16.4.15 `RCC_SysClk_TypeDef RCC_SysClkStatus ()`

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле `niietcm4_rcc.c` строка 394

8.16.4.16 `void RCC_UARTClkCmd (NT_UART_TypeDef * UARTx, FunctionalState State)`

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 526

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

8.16.4.17 `void RCC_UARTClkDivConfig (NT_UART_TypeDef * UARTx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState ↵.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 488

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_UART_ALL_PERIPH`↵.

8.16.4.18 void RCC_UARTClkSel (NT_UART_TypeDef * UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)

Настройка источника тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
RCC_UART↔Clk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 448

Перекрестные ссылки IS_RCC_UART_CLK и IS_UART_ALL_PERIPH.

8.16.4.19 void RCC_USBClkCmd (FunctionalState State)

Включение тактирования USB.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 425

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.16.4.20 void RCC_USBClkConfig (RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)

Настройка источника тактового сигнала для USB.

Аргументы

RCC_USBClk	Выбор источника тактирования. Параметр принимает любое значение из RCC_USBClk_TypeDef .
RCC_USB↔Freq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 408

Перекрестные ссылки IS_RCC_USB_CLK и IS_RCC_USB_FREQ.

8.17 Файл niietcm4_rtc.c

Файл содержит реализацию всех функции для работы с RTC.

```
#include "niietcm4_rtc.h"
```

Функции

- `uint32_t bcd2hex (uint32_t a)`
- `uint32_t hex2bcd (uint32_t x)`
- `void RTC_ShadowUpd (FunctionalState State)`
- `void RTC_GetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)`
- `void RTC_GetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)`
- `void RTC_SetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)`
- `void RTC_SetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)`

8.17.1 Подробное описание

Файл содержит реализацию всех функций для работы с RTC.

Автор

НИИЭТ

- Александр Дыхно (DAV), dykhno@niiet.ru
- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

04.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.18 Файл `niietcm4_rtc.h`

Файл содержит все прототипы функций для таймеров

```
#include "niietcm4.h"
```


Структуры данных

- struct `RTC_Time_TypeDef`
Структура времени.
- struct `RTC_Date_TypeDef`
Структура даты.

Макросы

- `#define IS_RTC_PSECOND(PSECOND) ((PSECOND) <= 0x3FF)`
Макрос проверки попадания значений долей секунд в допустимый диапазон.
- `#define IS_RTC_SECOND(SECOND) ((SECOND) <= 59)`
Макрос проверки попадания значений секунд в допустимый диапазон.
- `#define IS_RTC_MINUTE(MINUTE) ((MINUTE) <= 59)`
Макрос проверки попадания значений минут в допустимый диапазон.
- `#define IS_RTC_HOUR(HOUR) ((HOUR) <= 23)`
Макрос проверки попадания значений часов в допустимый диапазон.
- `#define IS_RTC_WEEKDAY(WEEKDAY)`
Макрос проверки аргументов типа `RTC_Weekday_TypeDef`.
- `#define IS_RTC_DAY(DAY) (((DAY) > 0) && ((DAY) <= 31))`
Макрос проверки попадания значений дней в допустимый диапазон.
- `#define IS_RTC_MONTH(MONTH)`
Макрос проверки аргументов типа `RTC_Month_TypeDef`.
- `#define IS_RTC_YEAR(YEAR) ((YEAR) <= 99)`
Макрос проверки попадания значений лет в допустимый диапазон.
- `#define IS_RTC_FORMAT(FORMAT)`
Макрос проверки аргументов типа `RTC_Format_TypeDef`.

Перечисления

- enum `RTC_Weekday_TypeDef` {
`RTC_Weekday_Monday = ((uint32_t)0x01)`, `RTC_Weekday_Tuesday = ((uint32_t)0x02)`, `RTC_Weekday_Wednesday = ((uint32_t)0x03)`, `RTC_Weekday_Thursday = ((uint32_t)0x04)`,
`RTC_Weekday_Friday = ((uint32_t)0x05)`, `RTC_Weekday_Saturday = ((uint32_t)0x06)`, `RTC_Weekday_Sunday = ((uint32_t)0x07)` }
Дни недели.
- enum `RTC_Month_TypeDef` {
`RTC_Month_January = ((uint32_t)0x01)`, `RTC_Month_February = ((uint32_t)0x02)`, `RTC_Month_March = ((uint32_t)0x03)`, `RTC_Month_April = ((uint32_t)0x04)`,
`RTC_Month_May = ((uint32_t)0x05)`, `RTC_Month_June = ((uint32_t)0x06)`, `RTC_Month_July = ((uint32_t)0x07)`, `RTC_Month_August = ((uint32_t)0x08)`,
`RTC_Month_September = ((uint32_t)0x09)`, `RTC_Month_October = ((uint32_t)0x0A)`, `RTC_Month_November = ((uint32_t)0x0B)`, `RTC_Month_December = ((uint32_t)0x0C)` }
Месяцы.
- enum `RTC_Format_TypeDef` { `RTC_Format_BIN`, `RTC_Format_BCD` }
Формат ввода/вывода времени и даты.

Функции

- void `RTC_GetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)`
- void `RTC_GetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)`
- void `RTC_SetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)`
- void `RTC_SetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)`

8.18.1 Подробное описание

Файл содержит все прототипы функций для таймеров

Автор

НИИЭТ

- Александр Дыхно (DAV), dykhno@niiet.ru
- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

04.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.18.2 Макросы

8.18.2.1 #define IS_RTC_FORMAT(FORMAT)

Макроопределение:

```
((FORMAT) == RTC_Format_BIN) || \
((FORMAT) == RTC_Format_BCD))
```

Макрос проверки аргументов типа `RTC_Format_TypeDef`.

См. определение в файле `niietcm4_rtc.h` строка 170

8.18.2.2 #define IS_RTC_MONTH(MONTH)

Макроопределение:

```
((MONTH) == RTC_Month_January) || \
((MONTH) == RTC_Month_February) || \
((MONTH) == RTC_Month_March) || \
((MONTH) == RTC_Month_April) || \
((MONTH) == RTC_Month_May) || \
((MONTH) == RTC_Month_June) || \
((MONTH) == RTC_Month_July) || \
```

```
((MONTH) == RTC_Month_August) || \
((MONTH) == RTC_Month_September) || \
((MONTH) == RTC_Month_October) || \
((MONTH) == RTC_Month_November) || \
((MONTH) == RTC_Month_December))
```

Макрос проверки аргументов типа `RTC_Month_TypeDef`.

См. определение в файле niietcm4_rtc.h строка 136

8.18.2.3 #define IS_RTC_WEEKDAY(WEEKDAY)

Макроопределение:

```
((WEEKDAY) == RTC_Weekday_Monday) || \
((WEEKDAY) == RTC_Weekday_Tuesday) || \
((WEEKDAY) == RTC_Weekday_Wednesday) || \
((WEEKDAY) == RTC_Weekday_Thursday) || \
((WEEKDAY) == RTC_Weekday_Friday) || \
((WEEKDAY) == RTC_Weekday_Saturday) || \
((WEEKDAY) == RTC_Weekday_Sunday))
```

Макрос проверки аргументов типа `RTC_Weekday_TypeDef`.

См. определение в файле niietcm4_rtc.h строка 97

8.18.3 Перечисления

8.18.3.1 enum RTC_Format_TypeDef

Формат ввода/вывода времени и даты.

Элементы перечислений

```
RTC_Format_BIN    Бинарный формат
RTC_Format_BCD    Двоично-десятичный формат
```

См. определение в файле niietcm4_rtc.h строка 159

8.18.3.2 enum RTC_Month_TypeDef

Месяцы.

Элементы перечислений

```
RTC_Month_January  January
RTC_Month_February February
RTC_Month_March    March
RTC_Month_April    April
RTC_Month_May      May
RTC_Month_June     June
RTC_Month_July     July
RTC_Month_August   August
RTC_Month_September September
RTC_Month_October  October
RTC_Month_November November
RTC_Month_December December
```

См. определение в файле niietcm4_rtc.h строка 115

8.18.3.3 enum RTC_Weekday_TypeDef

Дни недели.

Элементы перечислений

```
RTC_Weekday_Monday Monday
RTC_Weekday_Tuesday Tuesday
RTC_Weekday_Wednesday Wednesday
RTC_Weekday_Thursday Thursday
RTC_Weekday_Friday Friday
RTC_Weekday_Saturday Saturday
RTC_Weekday_Sunday Sunday
```

См. определение в файле niietcm4_rtc.h строка 81

8.19 Файл niietcm4_timer.c

Файл содержит реализацию всех функций для работы с таймерами

```
#include "niietcm4_timer.h"
```

Функции

- void [TIMER_Cmd](#) (NT_TIMER_TypeDef *TIMERx, [FunctionalState](#) State)
Разрешение работы выбранного таймера.
- void [TIMER_PeriodConfig](#) (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)
Настройка периода опустошения выбранного таймера.
- void [TIMER_FreqConfig](#) (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)
Настройка частоты опустошения выбранного таймера.
- void [TIMER_SetReload](#) (NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)
Установка значения перезагрузки.
- uint32_t [TIMER_GetReload](#) (NT_TIMER_TypeDef *TIMERx)
Получение текущего значения перезагрузки.
- void [TIMER_SetCounter](#) (NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)
Установка значения счетчика.
- uint32_t [TIMER_GetCounter](#) (NT_TIMER_TypeDef *TIMERx)
Получение текущего значения счетчика.
- void [TIMER_ExtInputConfig](#) (NT_TIMER_TypeDef *TIMERx, [TIMER_ExtInput_TypeDef](#) TIMER_ExtInput)
Выбор режима работы входа внешнего тактирования.
- void [TIMER_ITCmd](#) (NT_TIMER_TypeDef *TIMERx, [FunctionalState](#) State)
Разрешение работы прерывания выбранного таймера.
- [FlagStatus](#) [TIMER_ITStatus](#) (NT_TIMER_TypeDef *TIMERx)
Чтение статуса прерывания выбранного таймера.
- void [TIMER_ITStatusClear](#) (NT_TIMER_TypeDef *TIMERx)
Очищение статусного бита прерывания выбранного таймера.

8.19.1 Подробное описание

Файл содержит реализацию всех функции для работы с таймерами

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

03.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.19.2 Функции

8.19.2.1 void TIMER_Cmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 65

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

8.19.2.2 void TIMER_ExtInputConfig (NT_TIMER_TypeDef * TIMERx,
TIMER_ExtInput_TypeDef TIMER_ExtInput)

Выбор режима работы входа внешнего тактирования.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TIMER_Ext↔ Input	Выбор режима работы. Параметр принимает любое значение из TIMER_Ext↔ Input_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 171

Перекрестные ссылки IS_TIMER_ALL_PERIPH, IS_TIMER_EXT_INPUT, TIMER_ExtInput↔_CountClk и TIMER_ExtInput_CountEn.

8.19.2.3 void TIMER_FreqConfig (NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)

Настройка частоты опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемого периода опустошения таймера [TIMER_PeriodConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerFreq	Частота опустошения таймера в Гц.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 102

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.4 uint32_t TIMER_GetCounter (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_timer.c строка 156

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.5 uint32_t TIMER_GetReload (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_timer.c строка 129

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.6 void TIMER_ITCmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

8.19.2.7 FlagStatus TIMER_ITStatus (NT_TIMER_TypeDef * TIMERx)

Чтение статуса прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_timer.c строка 214

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.8 void TIMER_ITStatusClear (NT_TIMER_TypeDef * TIMERx)

Очищение статусного бита прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 238

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.9 void TIMER_PeriodConfig (NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)

Настройка периода опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемой частоты опустошения таймера [TIMER_FreqConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerPeriod	Период опустошения таймера в мкс.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 84

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.10 void TIMER_SetCounter (NT_TIMER_TypeDef * TIMERx, uint32_t CounterVal)

Установка значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
CounterVal	Значение счетчика.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 143

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19.2.11 void TIMER_SetReload (NT_TIMER_TypeDef * TIMERx, uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
ReloadVal	Значение перезагрузки.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 116

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.20 Файл niietcm4_timer.h

Файл содержит все прототипы функций для таймеров


```
#include "niietcm4.h"
```

Макросы

- `#define IS_TIMER_EXT_INPUT(EXT_INPUT)`
Макрос проверки аргументов типа `TIMER_ExtInput_TypeDef`.

Перечисления

- `enum TIMER_ExtInput_TypeDef { TIMER_ExtInput_Disable, TIMER_ExtInput_CountClk, TIMER_ExtInput_CountEn }`
Настройка внешнего тактирования таймера.

Функции

- `void TIMER_Cmd (NT_TIMER_TypeDef *TIMERx, FunctionalState State)`
Разрешение работы выбранного таймера.
- `void TIMER_PeriodConfig (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)`
Настройка периода опустошения выбранного таймера.
- `void TIMER_FreqConfig (NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)`
Настройка частоты опустошения выбранного таймера.
- `void TIMER_SetReload (NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)`
Установка значения перезагрузки.
- `uint32_t TIMER_GetReload (NT_TIMER_TypeDef *TIMERx)`
Получение текущего значения перезагрузки.
- `void TIMER_SetCounter (NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)`
Установка значения счетчика.
- `uint32_t TIMER_GetCounter (NT_TIMER_TypeDef *TIMERx)`
Получение текущего значения счетчика.
- `void TIMER_ExtInputConfig (NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)`
Выбор режима работы входа внешнего тактирования.
- `void TIMER_ITCmd (NT_TIMER_TypeDef *TIMERx, FunctionalState State)`
Разрешение работы прерывания выбранного таймера.
- `FlagStatus TIMER_ITStatus (NT_TIMER_TypeDef *TIMERx)`
Чтение статуса прерывания выбранного таймера.
- `void TIMER_ITStatusClear (NT_TIMER_TypeDef *TIMERx)`
Очищение статусного бита прерывания выбранного таймера.

8.20.1 Подробное описание

Файл содержит все прототипы функций для таймеров

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

03.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.20.2 Макросы

8.20.2.1 #define IS_TIMER_EXT_INPUT(EXT_INPUT)

Макроопределение:

```
((EXT_INPUT) == TIMER_ExtInput_Disable) || \
    ((EXT_INPUT) == TIMER_ExtInput_CountClk) || \
    ((EXT_INPUT) == TIMER_ExtInput_CountEn))
```

Макрос проверки аргументов типа `TIMER_ExtInput_TypeDef`.

См. определение в файле `niietcm4_timer.h` строка 67

Используется в `TIMER_ExtInputConfig()`.

8.20.3 Перечисления

8.20.3.1 enum TIMER_ExtInput_TypeDef

Настройка внешнего тактирования таймера.

Элементы перечислений

`TIMER_ExtInput_Disable` Внешнее тактирование не используется.

`TIMER_ExtInput_CountClk` Таймер считает по внешнему тактовому сигналу.

`TIMER_ExtInput_CountEn` Таймер считает по внутреннему тактовому сигналу и только тогда, когда на выводе "1".

См. определение в файле `niietcm4_timer.h` строка 56

8.20.4 Функции

8.20.4.1 void TIMER_Cmd (NT_TIMER_TypeDef * TIMEx, FunctionalState State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 65

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_TIMER_ALL_PERIPH`.

```
8.20.4.2 void TIMER_ExtInputConfig ( NT_TIMER_TypeDef * TIMERx,
    TIMER_ExtInput_TypeDef TIMER_ExtInput )
```

Выбор режима работы входа внешнего тактирования.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TIMER_ExtInput	Выбор режима работы. Параметр принимает любое значение из TIMER_ExtInput_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 171

Перекрестные ссылки `IS_TIMER_ALL_PERIPH`, `IS_TIMER_EXT_INPUT`, `TIMER_ExtInputCountClk` и `TIMER_ExtInput_CountEn`.

```
8.20.4.3 void TIMER_FreqConfig ( NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq,
    uint32_t TimerFreq )
```

Настройка частоты опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемого периода опустошения таймера [TIMER_PeriodConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerFreq	Частота опустошения таймера в Гц.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 102

Перекрестные ссылки `IS_TIMER_ALL_PERIPH`.

```
8.20.4.4 uint32_t TIMER_GetCounter ( NT_TIMER_TypeDef * TIMERx )
```

Получение текущего значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_timer.c строка 156

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.20.4.5 uint32_t TIMER_GetReload (NT_TIMER_TypeDef * TIMERx)

Получение текущего значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_timer.c строка 129

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.20.4.6 void TIMER_ITCmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

8.20.4.7 FlagStatus TIMER_ITStatus (NT_TIMER_TypeDef * TIMERx)

Чтение статуса прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_timer.c строка 214

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.20.4.8 void TIMER_ITStatusClear (NT_TIMER_TypeDef * TIMERx)

Очищение статусного бита прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 238

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

```
8.20.4.9 void TIMER_PeriodConfig ( NT_TIMER_TypeDef * TIMERx, uint32_t TimerClkFreq,
uint32_t TimerPeriod )
```

Настройка периода опустошения выбранного таймера.

Внимание

В качестве альтернативы может применяться как ручное заполнение регистра перезагрузки [TIMER_SetReload](#) так и автоматический расчет, исходя из желаемой частоты опустошения таймера [TIMER_FreqConfig](#).

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TimerClkFreq	Частота в Гц, которой тактируется таймер.
TimerPeriod	Период опустошения таймера в мкс.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 84

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

```
8.20.4.10 void TIMER_SetCounter ( NT_TIMER_TypeDef * TIMERx, uint32_t CounterVal )
```

Установка значения счетчика.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
CounterVal	Значение счетчика.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 143

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

```
8.20.4.11 void TIMER_SetReload ( NT_TIMER_TypeDef * TIMERx, uint32_t ReloadVal )
```

Установка значения перезагрузки.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
ReloadVal	Значение перезагрузки.

Возвращаемые значения

Нет

См. определение в файле niietcm4_timer.c строка 116

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.21 Файл niietcm4_uart.c

Файл содержит реализацию всех функции для работы с модулями UART.

```
#include "niietcm4_uart.h"
```

Функции

- void [UART_Cmd](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Разрешение работы выбранного UART.
- void [UART_BaudRateDivConfig](#) (NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)
Ручная настройка делителя для реализации необходимой скорости передачи.
- void [UART_Break](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Включение разрыва линии.
- void [UART_DeInit](#) (NT_UART_TypeDef *UARTx)
Устанавливает все регистры UART значениями по умолчанию.
- [OperationStatus](#) [UART_Init](#) (NT_UART_TypeDef *UARTx, [UART_Init_TypeDef](#) *UART_InitStruct)
Инициализирует UARTx согласно параметрам структуры UART_InitStruct.
- void [UART_StructInit](#) ([UART_Init_TypeDef](#) *UART_InitStruct)
Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.
- void [UART_SendData](#) (NT_UART_TypeDef *UARTx, uint32_t Data)
Передача слова данных.
- uint32_t [UART_RecieveData](#) (NT_UART_TypeDef *UARTx)
Прием слова данных.
- [FlagStatus](#) [UART_FlagStatus](#) (NT_UART_TypeDef *UARTx, [UART_Flag_Typedef](#) UART_Flag)
Запрос состояния выбранного флага.
- [FlagStatus](#) [UART_ErrorStatus](#) (NT_UART_TypeDef *UARTx, [UART_Error_Typedef](#) UART_Error)
Запрос состояния выбранного флага ошибки.
- void [UART_ErrorStatusClear](#) (NT_UART_TypeDef *UARTx)
Очистка флагов ошибки.
- void [UART_ModemConfig](#) (NT_UART_TypeDef *UARTx, [UART_ModemInit_TypeDef](#) *UART_ModemInitStruct)
Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.
- void [UART_ModemStructInit](#) ([UART_ModemInit_TypeDef](#) *UART_ModemInitStruct)
Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

- void `UART_ITFIFOLevelConfig` (NT_UART_TypeDef *UARTx, `UART_Dir_Typedef` UART_Dir, `UART_FIFOLevel_TypeDef` UART_FIFOLevel)
Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.
- void `UART_ITCmd` (NT_UART_TypeDef *UARTx, `UART_ITSource_Typedef` UART_ITSource, `FunctionalState` State)
Маскирование выбранных прерываний.
- `FlagStatus` `UART_ITRawStatus` (NT_UART_TypeDef *UARTx, `UART_ITSource_Typedef` UART_ITSource)
Запрос немаскированного состояния прерывания.
- `FlagStatus` `UART_ITMaskedStatus` (NT_UART_TypeDef *UARTx, `UART_ITSource_Typedef` UART_ITSource)
Запрос маскированного состояния прерывания.
- void `UART_ITStatusClear` (NT_UART_TypeDef *UARTx, `UART_ITSource_Typedef` UART_ITSource)
Сброс флагов состояния выбранных прерываний.
- void `UART_DMABlkOnErrCmd` (NT_UART_TypeDef *UARTx, `FunctionalState` State)
Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.
- void `UART_DMACmd` (NT_UART_TypeDef *UARTx, `UART_Dir_Typedef` UART_Dir, `FunctionalState` State)
Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

8.21.1 Подробное описание

Файл содержит реализацию всех функции для работы с модулями UART.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

18.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.21.2 Функции

8.21.2.1 void UART_BaudRateDivConfig (NT_UART_TypeDef * UARTx, uint32_t IntDiv, uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

8.21.2.2 void UART_Break (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.21.2.3 void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.21.2.4 void UART_DeInit (NT_UART_TypeDef * UARTx)

Устанавливает все регистры UART значениями по умолчанию.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 120

Перекрестные ссылки IS_UART_ALL_PERIPH, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_UART2, RCC_PeriphRst_UART3 и RCC_PeriphRstCmd().

8.21.2.5 void UART_DMABlkOnErrCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.21.2.6 void UART_DMACmd (NT_UART_TypeDef * UARTx, UART_Dir_Typedef UART_Dir, FunctionalState State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DIR и UART_Dir_Rx.

8.21.2.7 FlagStatus UART_ErrorStatus (NT_UART_TypeDef * UARTx, UART_Error_Typedef UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error↔_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

8.21.2.8 void UART_ErrorStatusClear (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.21.2.9 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_TypeDef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

8.21.2.10 OperationStatus UART_Init (NT_UART_TypeDef * UARTx, UART_Init_TypeDef * UART_InitStruct)

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_Init↔Struct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле niietcm4_uart.c строка 159

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DATA_WIDTH, IS_UART_FIFO_LEVEL, IS_UART_PARITY_BIT, IS_UART_STOP_BIT, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_Init_TypeDef::UART_FIFOEn, UART_Init_TypeDef::UART_FIFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Even, UART_ParityBit_High, UART_ParityBit_Low, UART_ParityBit_Odd, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit и UART_Init_TypeDef::UART_TxEn.

8.21.2.11 void UART_ITCmd (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 410

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

8.21.2.12 void UART_ITFIFOLevelConfig (NT_UART_TypeDef * UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор между буфером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFOLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и UART_Dir_Rx.

8.21.2.13 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.21.2.14 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.21.2.15 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource↔_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

8.21.2.16 void UART_ModemConfig (NT_UART_TypeDef * UARTx,
UART_ModemInit_TypeDef * UART_ModemInitStruct)

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInit↔Struct.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

8.21.2.17 void UART_ModemStructInit (UART_ModemInit_TypeDef * UART_ModemInitStruct)

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_ModemInitStruct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
----------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

8.21.2.18 uint32_t UART_RecieveData (NT_UART_TypeDef * UARTx)

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.21.2.19 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

8.21.2.20 void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

Аргументы

UART_Init↵ Struct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
----------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 229

Перекрестные ссылки EXT_OSC_VALUE, UART_Init_TypeDef::UART_BaudRate, UART_Init↵
__TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_DataWidth_8, U↵
ART_Init_TypeDef::UART_FIFOEn, UART_FIFOLevel_1_2, UART_Init_TypeDef::UART_F↵
IFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit,
UART_ParityBit_Disable, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_↵
StopBit, UART_StopBit_1 и UART_Init_TypeDef::UART_TxEn.

8.22 Файл niietcm4_uart.h

Файл содержит все прототипы функций для UART.

```
#include "niietcm4.h"
```

Структуры данных

- struct [UART_ModemInit_TypeDef](#)
Структура инициализации модемного режима.
- struct [UART_Init_TypeDef](#)
Структура инициализации UART.

Макросы

- [#define IS_UART_INT_DIV\(INT_DIV\)](#) (((INT_DIV) > ((uint32_t)0x0)) && ((INT_DIV) < ((uint32_t)0x10000)))
Макрос проверки соответствия величины целой части делителя baudrate UART диапазону.
- [#define IS_UART_FRAC_DIV\(FRAC_DIV\)](#) ((FRAC_DIV) < ((uint32_t)0x40))
Макрос проверки соответствия величины дробной части делителя baudrate UART диапазону.
- [#define IS_UART_DATA\(DATA\)](#) ((DATA) < ((uint32_t)0x100))
Макрос проверки корректности передаваемых данных.
- [#define IS_UART_FLAG\(FLAG\)](#)
Макрос проверки аргументов типа [UART_Flag_TypeDef](#).
- [#define IS_UART_ERROR\(ERROR\)](#)
Макрос проверки аргументов типа [UART_Error_TypeDef](#).
- [#define IS_UART_IT_SOURCE\(IT_SOURCE\)](#) (((IT_SOURCE) > ((uint32_t)0x0)) && ((I↵
T_SOURCE) < ((uint32_t)0x800)))
Макрос проверки аргументов типа [UART_ITSource_TypeDef](#).
- [#define IS_UART_GET_IT_SOURCE\(IT_SOURCE\)](#)
Макрос проверки номера пина при работе с пинами по отдельности.
- [#define IS_UART_DIR\(DIR\)](#)
Макрос проверки аргументов типа [UART_Dir_TypeDef](#).
- [#define IS_UART_STOP_BIT\(STOP_BIT\)](#)
Макрос проверки аргументов типа [UART_StopBit_TypeDef](#).

- `#define IS_UART_PARITY_BIT(PARITY_BIT)`
Макрос проверки аргументов типа `UART_ParityBit_TypeDef`.
- `#define IS_UART_DATA_WIDTH(DATA_WIDTH)`
Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.
- `#define IS_UART_FIFO_LEVEL(FIFO_LEVEL)`
Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

Перечисления

- `enum UART_Flag_TypeDef {`
`UART_Flag_InvCTS, UART_Flag_InvDSR, UART_Flag_InvDCD, UART_Flag_Busy,`
`UART_Flag_RxFIFOEmpty, UART_Flag_TxFIFOFull, UART_Flag_RxFIFOFull, UART_↵`
`Flag_TxFIFOEmpty,`
`UART_Flag_InvRI }`
Перечень флагов.
- `enum UART_Error_TypeDef { UART_Error_Frame, UART_Error_Parity, UART_Error_↵`
`Break, UART_Error_Overflow }`
Перечень ошибок приемника.
- `enum UART_ITSource_TypeDef {`
`UART_ITSource_ChangeRI = ((uint32_t)0x00000001), UART_ITSource_ChangeCTS =`
`((uint32_t)0x00000002), UART_ITSource_ChangeDCD = ((uint32_t)0x00000004), UART_↵`
`ITSource_ChangeDSR = ((uint32_t)0x00000008),`
`UART_ITSource_RxFIFOLevel = ((uint32_t)0x00000010), UART_ITSource_TxFIFOLevel =`
`((uint32_t)0x00000020), UART_ITSource_RecieveTimeout = ((uint32_t)0x00000040), UART_↵`
`ITSource_ErrorFrame = ((uint32_t)0x00000080),`
`UART_ITSource_ErrorParity = ((uint32_t)0x00000100), UART_ITSource_ErrorBreak =`
`((uint32_t)0x00000200), UART_ITSource_ErrorOverflow = ((uint32_t)0x00000400) }`
Источники прерываний UART.
- `enum UART_Dir_TypeDef { UART_Dir_Rx, UART_Dir_Tx }`
Направления передачи UART.
- `enum UART_StopBit_TypeDef { UART_StopBit_1, UART_StopBit_2 }`
Выбор режима передачи стопового бита.
- `enum UART_ParityBit_TypeDef {`
`UART_ParityBit_Disable, UART_ParityBit_Odd, UART_ParityBit_Even, UART_Parity_↵`
`Bit_High,`
`UART_ParityBit_Low }`
Выбор режима бита четности.
- `enum UART_DataWidth_TypeDef { UART_DataWidth_5, UART_DataWidth_6, UART_↵`
`DataWidth_7, UART_DataWidth_8 }`
Количество передаваемых/принимаемых информационных бит.
- `enum UART_FIFOLevel_TypeDef {`
`UART_FIFOLevel_1_8, UART_FIFOLevel_1_4, UART_FIFOLevel_1_2, UART_FIFO_↵`
`Level_3_4,`
`UART_FIFOLevel_7_8 }`
Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

Функции

- `void UART_Cmd (NT_UART_TypeDef *UARTx, FunctionalState State)`
Разрешение работы выбранного UART.
- `void UART_BaudRateDivConfig (NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t_↵`
`t FracDiv)`

- Ручная настройка делителя для реализации необходимой скорости передачи.

 - void **UART_Break** (NT_UART_TypeDef *UARTx, **FunctionalState** State)

Включение разрыва линии.
- void **UART_DeInit** (NT_UART_TypeDef *UARTx)

Устанавливает все регистры UART значениями по умолчанию.
- **OperationStatus** **UART_Init** (NT_UART_TypeDef *UARTx, **UART_Init_TypeDef** *UART_InitStruct)

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.
- void **UART_StructInit** (**UART_Init_TypeDef** *UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.
- void **UART_SendData** (NT_UART_TypeDef *UARTx, uint32_t Data)

Передача слова данных.
- uint32_t **UART_RecieveData** (NT_UART_TypeDef *UARTx)

Прием слова данных.
- **FlagStatus** **UART_FlagStatus** (NT_UART_TypeDef *UARTx, **UART_Flag_Typedef** UART_Flag)

Запрос состояния выбранного флага.
- **FlagStatus** **UART_ErrorStatus** (NT_UART_TypeDef *UARTx, **UART_Error_Typedef** UART_Error)

Запрос состояния выбранного флага ошибки.
- void **UART_ErrorStatusClear** (NT_UART_TypeDef *UARTx)

Очистка флагов ошибки.
- void **UART_ModemConfig** (NT_UART_TypeDef *UARTx, **UART_ModemInit_TypeDef** *UART_ModemInitStruct)

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.
- void **UART_ModemStructInit** (**UART_ModemInit_TypeDef** *UART_ModemInitStruct)

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.
- void **UART_ITFIFOLevelConfig** (NT_UART_TypeDef *UARTx, **UART_Dir_Typedef** UART_Dir, **UART_FIFOLevel_TypeDef** UART_FIFOLevel)

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.
- void **UART_ITCmd** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource, **FunctionalState** State)

Маскирование выбранных прерываний.
- **FlagStatus** **UART_ITRawStatus** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource)

Запрос немаскированного состояния прерывания.
- **FlagStatus** **UART_ITMaskedStatus** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource)

Запрос маскированного состояния прерывания.
- void **UART_ITStatusClear** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource)

Сброс флагов состояния выбранных прерываний.
- void **UART_DMABlkOnErrCmd** (NT_UART_TypeDef *UARTx, **FunctionalState** State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.
- void **UART_DMACmd** (NT_UART_TypeDef *UARTx, **UART_Dir_Typedef** UART_Dir, **FunctionalState** State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

8.22.1 Подробное описание

Файл содержит все прототипы функций для UART.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

18.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.22.2 Макросы

8.22.2.1 #define IS_UART_DATA_WIDTH(DATA_WIDTH)

Макроопределение:

```
((DATA_WIDTH) == UART_DataWidth_5) || \
((DATA_WIDTH) == UART_DataWidth_6) || \
((DATA_WIDTH) == UART_DataWidth_7) || \
((DATA_WIDTH) == UART_DataWidth_8))
```

Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.

См. определение в файле `niietcm4_uart.h` строка 236

Используется в `UART_Init()`.

8.22.2.2 #define IS_UART_DIR(DIR)

Макроопределение:

```
((DIR) == UART_Dir_Rx) || \
((DIR) == UART_Dir_Tx))
```

Макрос проверки аргументов типа `UART_Dir_TypeDef`.

См. определение в файле `niietcm4_uart.h` строка 177

Используется в `UART_DMAMCmd()` и `UART_ITFIFOLevelConfig()`.

8.22.2.3 #define IS_UART_ERROR(ERROR)

Макроопределение:

```
((ERROR) == UART_Error_Frame) || \
((ERROR) == UART_Error_Parity) || \
((ERROR) == UART_Error_Break) || \
((ERROR) == UART_Error_Overflow)
```

Макрос проверки аргументов типа [UART_Error_Typedef](#).

См. определение в файле niietcm4_uart.h строка 117

Используется в `UART_ErrorStatus()`.

8.22.2.4 #define IS_UART_FIFO_LEVEL(FIFO_LEVEL)

Макроопределение:

```
((FIFO_LEVEL) == UART_FIFOLevel_1_8) || \
((FIFO_LEVEL) == UART_FIFOLevel_1_4) || \
((FIFO_LEVEL) == UART_FIFOLevel_1_2) || \
((FIFO_LEVEL) == UART_FIFOLevel_3_4) || \
((FIFO_LEVEL) == UART_FIFOLevel_7_8))
```

Макрос проверки аргументов типа [UART_FIFOLevel_TypeDef](#).

См. определение в файле niietcm4_uart.h строка 259

Используется в `UART_Init()` и `UART_ITFIFOLevelConfig()`.

8.22.2.5 #define IS_UART_FLAG(FLAG)

Макроопределение:

```
((FLAG) == UART_Flag_InvCTS) || \
((FLAG) == UART_Flag_InvDSR) || \
((FLAG) == UART_Flag_InvDCD) || \
((FLAG) == UART_Flag_Busy) || \
((FLAG) == UART_Flag_RxFIFOEmpty) || \
((FLAG) == UART_Flag_TxFIFOFull) || \
((FLAG) == UART_Flag_RxFIFOFull) || \
((FLAG) == UART_Flag_TxFIFOEmpty) || \
((FLAG) == UART_Flag_InvRI))
```

Макрос проверки аргументов типа [UART_Flag_Typedef](#).

См. определение в файле niietcm4_uart.h строка 91

Используется в `UART_FlagStatus()`.

8.22.2.6 #define IS_UART_GET_IT_SOURCE(IT_SOURCE)

Макроопределение:

```
((IT_SOURCE) == UART_ITSource_ChangeRI) || \
((IT_SOURCE) == \
UART_ITSource_ChangeCTS) || \
((IT_SOURCE) == \
UART_ITSource_ChangeDCD) || \
((IT_SOURCE) == \
UART_ITSource_ChangeDSR) || \
((IT_SOURCE) == \
UART_ITSource_RxFIFOLevel) || \
((IT_SOURCE) == \
UART_ITSource_TxFIFOLevel) || \
((IT_SOURCE) == \
UART_ITSource_RecieveTimeout) || \
```

```

        ((IT_SOURCE) ==
UART_ITSource_ErrorFrame) || \
        ((IT_SOURCE) ==
UART_ITSource_ErrorParity) || \
        ((IT_SOURCE) ==
UART_ITSource_ErrorBreak) || \
        ((IT_SOURCE) ==
UART_ITSource_ErrorOverflow))

```

Макрос проверки номера пина при работе с пинами по отдельности.

См. определение в файле niietcm4_uart.h строка 151

Используется в UART_ITMaskedStatus() и UART_ITRawStatus().

8.22.2.7 #define IS_UART_PARITY_BIT(PARITY_BIT)

Макроопределение:

```

(((PARITY_BIT) == UART_ParityBit_Disable) || \
 ((PARITY_BIT) == UART_ParityBit_Odd) || \
 ((PARITY_BIT) == UART_ParityBit_Even) || \
 ((PARITY_BIT) == UART_ParityBit_High) || \
 ((PARITY_BIT) == UART_ParityBit_Low))

```

Макрос проверки аргументов типа UART_ParityBit_TypeDef.

См. определение в файле niietcm4_uart.h строка 214

Используется в UART_Init().

8.22.2.8 #define IS_UART_STOP_BIT(STOP_BIT)

Макроопределение:

```

(((STOP_BIT) == UART_StopBit_1) || \
 ((STOP_BIT) == UART_StopBit_2))

```

Макрос проверки аргументов типа UART_StopBit_TypeDef.

См. определение в файле niietcm4_uart.h строка 194

Используется в UART_Init().

8.22.3 Перечисления

8.22.3.1 enum UART_DataWidth_TypeDef

Количество передаваемых/принимаемых информационных бит.

Элементы перечислений

UART_DataWidth_5 Длина информационного слова 5 бит.

UART_DataWidth_6 Длина информационного слова 6 бит.

UART_DataWidth_7 Длина информационного слова 7 бит.

UART_DataWidth_8 Длина информационного слова 8 бит.

См. определение в файле niietcm4_uart.h строка 224

8.22.3.2 enum UART_Dir_Typedef

Направления передачи UART.

Элементы перечислений

UART_Dir_Rx Передача.

UART_Dir_Tx Прием.

См. определение в файле niietcm4_uart.h строка 167

8.22.3.3 enum UART_Error_Typedef

Перечень ошибок приемника.

Элементы перечислений

UART_Error_Frame Флаг ошибки в структуре кадра.

UART_Error_Parity Флаг ошибки контроля четности.

UART_Error_Break Флаг разрыва линии.

UART_Error_Overflow Флаг переполнения буфера приемника.

См. определение в файле niietcm4_uart.h строка 105

8.22.3.4 enum UART_FIFOLevel_TypeDef

Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

Элементы перечислений

UART_FIFOLevel_1_8 Заполнение FIFO на 1/8.

UART_FIFOLevel_1_4 Заполнение FIFO на 1/4.

UART_FIFOLevel_1_2 Заполнение FIFO на 1/2.

UART_FIFOLevel_3_4 Заполнение FIFO на 3/4.

UART_FIFOLevel_7_8 Заполнение FIFO на 7/8.

См. определение в файле niietcm4_uart.h строка 246

8.22.3.5 enum UART_Flag_Typedef

Перечень флагов.

Элементы перечислений

UART_Flag_InvCTS Флаг инверсии сигнала на линии UART_CTS.

UART_Flag_InvDSR Флаг инверсии сигнала на линии UART_DSR.

UART_Flag_InvDCD Флаг инверсии сигнала на линии UART_DSR.

UART_Flag_Busy Флаг занятости блока UART.

UART_Flag_RxFIFOEmpty Флаг пустоты буфера приемника.

UART_Flag_TxFIFOFull Флаг заполнения буфера передатчика.

UART_Flag_RxFIFOFull Флаг заполнения буфера приемника.

UART_Flag_TxFIFOEmpty Флаг пустоты буфера передатчика.

UART_Flag_InvRI Флаг инверсии сигнала на линии UART_RI.

См. определение в файле niietcm4_uart.h строка 74

8.22.3.6 enum UART_ITSource_TypeDef

Источники прерываний UART.

Элементы перечислений

UART_ITSource_ChangeRI Изменение состояния линии UART_RI
 UART_ITSource_ChangeCTS Изменение состояния линии UART_CTS
 UART_ITSource_ChangeDCD Изменение состояния линии UART_DCD
 UART_ITSource_ChangeDSR Изменение состояния линии UART_DSR
 UART_ITSource_RxFIFOLevel Порог переполнения буфера приемника
 UART_ITSource_TxFIFOLevel Порог опустошения буфера передатчика
 UART_ITSource_RecieveTimeout Таймаут приема данных
 UART_ITSource_ErrorFrame Ошибка в структуре кадра
 UART_ITSource_ErrorParity Ошибка контроля четности
 UART_ITSource_ErrorBreak Разрыв линии
 UART_ITSource_ErrorOverflow Переполнение буфера приемника

См. определение в файле niietcm4_uart.h строка 126

8.22.3.7 enum UART_ParityBit_TypeDef

Выбор режима бита четности.

Элементы перечислений

UART_ParityBit_Disable Не передается, не проверяется.
 UART_ParityBit_Odd Проверка нечетности данных.
 UART_ParityBit_Even Проверка четности данных.
 UART_ParityBit_High Бит четности постоянно равен единице.
 UART_ParityBit_Low Бит четности постоянно равен нулю.

См. определение в файле niietcm4_uart.h строка 201

8.22.3.8 enum UART_StopBit_TypeDef

Выбор режима передачи стопового бита.

Элементы перечислений

UART_StopBit_1 Один стоповый бит.
 UART_StopBit_2 Два стоповых бита.

См. определение в файле niietcm4_uart.h строка 184

8.22.4 Функции

8.22.4.1 void UART_BaudRateDivConfig (NT_UART_TypeDef * UARTx, uint32_t IntDiv, uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

8.22.4.2 void UART_Break (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.22.4.3 void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.22.4.4 void UART_DeInit (NT_UART_TypeDef * UARTx)

Устанавливает все регистры UART значениями по умолчанию.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
-------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 120

Перекрестные ссылки IS_UART_ALL_PERIPH, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_UART2, RCC_PeriphRst_UART3 и RCC_PeriphRstCmd().

8.22.4.5 void UART_DMABlkOnErrCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.22.4.6 void UART_DMACmd (NT_UART_TypeDef * UARTx, UART_Dir_Typedef UART_Dir, FunctionalState State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DIR и UART_Dir_Rx.

8.22.4.7 FlagStatus UART_ErrorStatus (NT_UART_TypeDef * UARTx, UART_Error_Typedef UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

8.22.4.8 void UART_ErrorStatusClear (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.22.4.9 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_Typedef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

8.22.4.10 OperationStatus UART_Init (NT_UART_TypeDef * UARTx, UART_Init_TypeDef * UART_InitStruct)

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле niietcm4_uart.c строка 159

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DATA_WIDTH, IS_UART_FIFO_LEVEL, IS_UART_PARITY_BIT, IS_UART_STOP_BIT, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_Init_TypeDef::UART_FIFOEn, UART_Init_TypeDef::UART_F

IFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Even, UART_ParityBit_High, UART_ParityBit_Low, UART_ParityBit_Odd, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit и UART_Init_TypeDef::UART_TxEn.

8.22.4.11 void UART_ITCmd (NT_UART_TypeDef * UARTx, UART_ITSource_Typedef UART_ITSource, FunctionalState State)

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 410

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

8.22.4.12 void UART_ITFIFOLevelConfig (NT_UART_TypeDef * UARTx, UART_Dir_Typedef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор между буффером приемника и передатчика. Параметр принимает любое из значений UART_Dir_Typedef .
UART_FIFOLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и UART_Dir_Rx.

8.22.4.13 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx, UART_ITSource_Typedef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерывания. Параметр принимает любое значение из UART_ITSource_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.22.4.14 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.22.4.15 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

8.22.4.16 void UART_ModemConfig (NT_UART_TypeDef * UARTx,
UART_ModemInit_TypeDef * UART_ModemInitStruct)

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInit↔
Struct.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
--------------------------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

8.22.4.17 void UART_ModemStructInit (UART_ModemInit_TypeDef * UART_ModemInitStruct)

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_ModemInitStruct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
----------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

8.22.4.18 uint32_t UART_RecieveData (NT_UART_TypeDef * UARTx)

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.22.4.19 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

8.22.4.20 void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

Аргументы

UART_Init↵ Struct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
----------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 229

Перекрестные ссылки EXT_OSC_VALUE, UART_Init_TypeDef::UART_BaudRate, UART_Init↵
_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_DataWidth_8, U↵
ART_Init_TypeDef::UART_FIFOEn, UART_FIFOLevel_1_2, UART_Init_TypeDef::UART_F↵
IFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit,
UART_ParityBit_Disable, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_↵
StopBit, UART_StopBit_1 и UART_Init_TypeDef::UART_TxEn.

8.23 Файл niietcm4_userflash.c

Файл содержит реализацию всех функции для работы с пользовательской флеш.

```
#include "niietcm4_userflash.h"
```

Функции

- void [USERFLASH_Init](#) (uint32_t SysClkFreq)
Инициализирует тайминги доступа для контроллера пользовательской флеш.
- [USERFLASH_Status_TypeDef USERFLASH_OperationStatus](#) ()
Статус работы контроллера пользовательской флеш.
- void [USERFLASH_OperationStatusClear](#) ()
Очищает статус работы контроллера пользовательской флеш.
- void [USERFLASH_FullErase](#) ()
Полная очистка основной области пользовательской флеш.
- uint32_t [USERFLASH_Read](#) (uint32_t Address)
Чтение байта из основной области пользовательской флеш.
- void [USERFLASH_Write](#) (uint32_t Address, uint32_t Data)
Запись байта в основную область пользовательской флеш по указанному адресу.
- void [USERFLASH_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы основной области пользовательской флеш.
- uint32_t [USERFLASH_Info_Read](#) (uint32_t Address)
Чтение байта из информационной области пользовательской флеш.
- void [USERFLASH_Info_Write](#) (uint32_t Address, uint32_t Data)
Запись байта в информационную область пользовательской флеш по указанному адресу.
- void [USERFLASH_Info_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы информационной области пользовательской флеш.
- void [USERFLASH_ITCmd](#) (FunctionalState State)
Включение прерывания по завершению чтения/записи/стирания.

8.23.1 Подробное описание

Файл содержит реализацию всех функции для работы с пользовательской флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

07.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.23.2 Функции

8.23.2.1 void USERFLASH_FullErase ()

Полная очистка основной области пользовательской флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 103

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OperationStatus() и USERFLASH_Status_None.

8.23.2.2 void USERFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области пользовательской флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 219

Перекрестные ссылки IS_USERFLASH_INFO_PAGE_NUM, USERFLASH_MAGIC_KEY и USERFLASH_PAGE_SIZE_BYTES.

8.23.2.3 uint32_t USERFLASH_Info_Read (uint32_t Address)

Чтение байта из информационной области пользовательской флеш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 175

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

8.23.2.4 void USERFLASH_Info_Write (uint32_t Address, uint32_t Data)

Запись байта в информационную область пользовательской флеш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 206

Перекрестные ссылки USERFLASH_MAGIC_KEY.

8.23.2.5 void USERFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера пользовательской флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
------------	---------------------------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 66

8.23.2.6 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.23.2.7 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из USERFLASH_↵_Status_TypeDef .
--------	---

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

8.23.2.8 void USERFLASH_OperationStatusClear ()

Очищает статус работы контроллера пользовательской флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 93

Используется в USERFLASH_Info_Read() и USERFLASH_Read().

8.23.2.9 void USERFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области пользовательской флэш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 161

Перекрестные ссылки IS_USERFLASH_PAGE_NUM, USERFLASH_MAGIC_KEY и USERFLASH_PAGE_SIZE_BYTES.

8.23.2.10 uint32_t USERFLASH_Read (uint32_t Address)

Чтение байта из основной области пользовательской флэш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 116

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

8.23.2.11 void USERFLASH_Write (uint32_t Address, uint32_t Data)

Запись байта в основную область пользовательской флэш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_userflash.c` строка 148

Перекрестные ссылки `USERFLASH_MAGIC_KEY`.

8.24 Файл `niietcm4_userflash.h`

Файл содержит все прототипы функций для пользовательской флеш.

```
#include "niietcm4.h"
```

Макросы

- `#define USERFLASH_OPERATION_TIMEOUT ((uint32_t)10000000)`
Время ожидания выполнения операции с флеш.
- `#define USERFLASH_MAGIC_KEY ((uint32_t)0xA4420000)`
Ключ для проведения операций с контроллером пользовательской флеш.
- `#define USERFLASH_PAGE_SIZE_BYTES ((uint32_t)256)`
- `#define USERFLASH_PAGE_TOTAL ((uint32_t)256)`
- `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`
- `#define IS_USERFLASH_PAGE_NUM(PAGE_NUM) (PAGE_NUM < USERFLASH_PAGE_TOTAL)`
Макрос проверки номера страницы основной области пользовательской флеш на попадание в допустимый диапазон.
- `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES`
- `#define USERFLASH_INFO_PAGE_TOTAL ((uint32_t)2)`
- `#define USERFLASH_INFO_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_INFO_PAGE_TOTAL)`
- `#define IS_USERFLASH_INFO_PAGE_NUM(PAGE_NUM) (PAGE_NUM < USERFLASH_INFO_PAGE_TOTAL)`
Макрос проверки номера страницы информационной области пользовательской флеш на попадание в допустимый диапазон.
- `#define IS_USERFLASH_STATUS(STATUS)`
Макрос проверки аргументов типа `USERFLASH_StatusTypeDef`.

Перечисления

- `enum USERFLASH_StatusTypeDef { USERFLASH_Status_None = ((uint32_t)0), USERFLASH_Status_Complete = ((uint32_t)1), USERFLASH_Status_Error = ((uint32_t)3) }`
Статус работы контроллера пользовательской флеш-памяти.

Функции

- `void USERFLASH_Init (uint32_t SysClkFreq)`
Инициализирует тайминги доступа для контроллера пользовательской флеш.

- `USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()`
Статус работы контроллера пользовательской флэш.
- `void USERFLASH_OperationStatusClear ()`
Очищает статус работы контроллера пользовательской флэш.
- `void USERFLASH_ITCmd (FunctionalState State)`
Включение прерывания по завершению чтения/записи/стирания.
- `uint32_t USERFLASH_Read (uint32_t Address)`
Чтение байта из основной области пользовательской флэш.
- `void USERFLASH_Write (uint32_t Address, uint32_t Data)`
Запись байта в основную область пользовательской флэш по указанному адресу.
- `void USERFLASH_PageErase (uint32_t PageNum)`
Стирание указанной страницы основной области пользовательской флэш.
- `void USERFLASH_FullErase ()`
Полная очистка основной области пользовательской флэш.
- `uint32_t USERFLASH_Info_Read (uint32_t Address)`
Чтение байта из информационной области пользовательской флэш.
- `void USERFLASH_Info_Write (uint32_t Address, uint32_t Data)`
Запись байта в информационную область пользовательской флэш по указанному адресу.
- `void USERFLASH_Info_PageErase (uint32_t PageNum)`
Стирание указанной страницы информационной области пользовательской флэш.

8.24.1 Подробное описание

Файл содержит все прототипы функций для пользовательской флэш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

07.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.24.2 Макросы

8.24.2.1 #define IS_USERFLASH_STATUS(STATUS)

Макроопределение:

```
((STATUS) == USERFLASH_Status_None) || \
    ((STATUS) == USERFLASH_Status_Complete) || \
    ((STATUS) == USERFLASH_Status_Error))
```

Макрос проверки аргументов типа `USERFLASH_Status_TypeDef`.

См. определение в файле `niietcm4_userflash.h` строка 123

8.24.2.2 #define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES

Размер страницы в байтах.

См. определение в файле `niietcm4_userflash.h` строка 86

8.24.2.3 #define USERFLASH_INFO_PAGE_TOTAL ((uint32_t)2)

Общее количество страниц.

См. определение в файле `niietcm4_userflash.h` строка 87

8.24.2.4 #define USERFLASH_INFO_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)

Общий размер информационной области.

См. определение в файле `niietcm4_userflash.h` строка 88

8.24.2.5 #define USERFLASH_PAGE_SIZE_BYTES ((uint32_t)256)

Размер страницы в байтах.

См. определение в файле `niietcm4_userflash.h` строка 68

Используется в `USERFLASH_Info_PageErase()` и `USERFLASH_PageErase()`.

8.24.2.6 #define USERFLASH_PAGE_TOTAL ((uint32_t)256)

Общее количество страниц.

См. определение в файле `niietcm4_userflash.h` строка 69

8.24.2.7 #define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)

Общий размер основной области.

См. определение в файле `niietcm4_userflash.h` строка 70

8.24.3 Перечисления

8.24.3.1 enum USERFLASH_Status_TypeDef

Статус работы контроллера пользовательской флеш-памяти.

Элементы перечислений

USERFLASH_Status_None Операция выполняется или отсутствует.

USERFLASH_Status_Complete Операция успешно завершена.

USERFLASH_Status_Error Операция завершена с ошибкой.

См. определение в файле niietcm4_userflash.h строка 112

8.24.4 Функции

8.24.4.1 void USERFLASH_FullErase ()

Полная очистка основной области пользовательской флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 103

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OperationStatus() и USERFLASH_Status_None.

8.24.4.2 void USERFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области пользовательской флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 219

Перекрестные ссылки IS_USERFLASH_INFO_PAGE_NUM, USERFLASH_MAGIC_KEY и USERFLASH_PAGE_SIZE_BYTES.

8.24.4.3 uint32_t USERFLASH_Info_Read (uint32_t Address)

Чтение байта из информационной области пользовательской флеш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 175

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

8.24.4.4 void USERFLASH_Info_Write (uint32_t Address, uint32_t Data)

Запись байта в информационную область пользовательской флеш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 206

Перекрестные ссылки USERFLASH_MAGIC_KEY.

8.24.4.5 void USERFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера пользовательской флеш.

Аргументы

SysClkFreq	Текущая системная частота в Гц.
------------	---------------------------------

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 66

8.24.4.6 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.24.4.7 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флэш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из USERFLASH← _Status_TypeDef .
--------	---

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

8.24.4.8 void USERFLASH_OperationStatusClear ()

Очищает статус работы контроллера пользовательской флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_userflash.c строка 93

Используется в USERFLASH_Info_Read() и USERFLASH_Read().

8.24.4.9 void USERFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области пользовательской флеш.

Аргументы

PageNum	Номер страницы.
---------	-----------------

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 161

Перекрестные ссылки IS_USERFLASH_PAGE_NUM, USERFLASH_MAGIC_KEY и USERFLASH_PAGE_SIZE_BYTES.

8.24.4.10 uint32_t USERFLASH_Read (uint32_t Address)

Чтение байта из основной области пользовательской флеш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 116

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

8.24.4.11 void USERFLASH_Write (uint32_t Address, uint32_t Data)

Запись байта в основную область пользовательской флеш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 148

Перекрестные ссылки USERFLASH_MAGIC_KEY.

8.25 Файл niietcm4_watchdog.c

Файл содержит реализацию всех функции для работы со сторожевым таймером.

```
#include "niietcm4_watchdog.h"
```

Макросы

- `#define WATCHDOG_Lock_Value ((uint32_t)0xDEADC0DE)`
- `#define WATCHDOG_Unlock_Value ((uint32_t)0x1ACCE551)`

Функции

- `void WATCHDOG_Cmd (FunctionalState State)`
Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.
- `void WATCHDOG_SetReload (uint32_t ReloadVal)`
Установка значения перезагрузки.
- `uint32_t WATCHDOG_GetReload ()`
Получение текущего значения перезагрузки.
- `uint32_t WATCHDOG_GetCounter ()`
Получение текущего значения счетчика.
- `void WATCHDOG_RstCmd (FunctionalState State)`
Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.
- `void WATCHDOG_LockCmd (FunctionalState State)`
Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.
- `FlagStatus WATCHDOG_ITRawStatus ()`
Чтение немаскированного флага прерывания сторожевого таймера.
- `FlagStatus WATCHDOG_ITMaskedStatus ()`
Чтение маскированного флага прерывания сторожевого таймера.
- `void WATCHDOG_ITStatusClear ()`
Очищение статусного бита прерывания сторожевого таймера.

8.25.1 Подробное описание

Файл содержит реализацию всех функций для работы со сторожевым таймером.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

15.01.2016

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2016 ОАО "НИИЭТ"

8.25.2 Макросы

8.25.2.1 #define WATCHDOG_Lock_Value ((uint32_t)0xDEADC0DE)

Любое значение для блокировки записи в регистры таймера

См. определение в файле niietcm4_watchdog.c строка 50

Используется в WATCHDOG_LockCmd().

8.25.2.2 #define WATCHDOG_Unlock_Value ((uint32_t)0x1ACCE551)

Значение для разблокировки записи в регистры таймера

См. определение в файле niietcm4_watchdog.c строка 51

Используется в WATCHDOG_LockCmd().

8.25.3 Функции

8.25.3.1 void WATCHDOG_Cmd (FunctionalState State)

Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.25.3.2 uint32_t WATCHDOG_GetCounter ()

Получение текущего значения счетчика.

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_watchdog.c строка 105

8.25.3.3 uint32_t WATCHDOG_GetReload ()

Получение текущего значения перезагрузки.

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_watchdog.c строка 95

8.25.3.4 FlagStatus WATCHDOG_ITMaskedStatus ()

Чтение маскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 174

8.25.3.5 FlagStatus WATCHDOG_ITRawStatus ()

Чтение немаскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 153

8.25.3.6 void WATCHDOG_ITStatusClear ()

Очищение статусного бита прерывания сторожевого таймера.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_watchdog.c строка 195

8.25.3.7 void WATCHDOG_LockCmd (FunctionalState State)

Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_watchdog.c строка 134

Перекрестные ссылки IS_FUNCTIONAL_STATE, WATCHDOG_Lock_Value и WATCHDOG_Unlock_Value.

8.25.3.8 void WATCHDOG_RstCmd (FunctionalState State)

Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 119

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.25.3.9 void WATCHDOG_SetReload (uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

ReloadVal	Значение перезагрузки. Параметр принимает любое значение из диапазона 0x1 - 0xFFFFFFFF.
-----------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 83

Перекрестные ссылки IS_WATCHDOG_RELOAD.

8.26 Файл niietcm4_watchdog.h

Файл содержит все прототипы функций для сторожевого таймера.

```
#include "niietcm4.h"
```

Макросы

- `#define IS_WATCHDOG_RELOAD(RELOAD) ((RELOAD) > ((uint32_t)0x0))`
Макрос проверки соответствия величины значения перезагрузки диапазону.

Функции

- void [WATCHDOG_Cmd](#) ([FunctionalState](#) State)
Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.
- void [WATCHDOG_SetReload](#) (uint32_t ReloadVal)
Установка значения перезагрузки.
- uint32_t [WATCHDOG_GetReload](#) ()
Получение текущего значения перезагрузки.
- uint32_t [WATCHDOG_GetCounter](#) ()
Получение текущего значения счетчика.
- void [WATCHDOG_RstCmd](#) ([FunctionalState](#) State)

Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.

- void [WATCHDOG_LockCmd](#) ([FunctionalState](#) State)
Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.
- [FlagStatus WATCHDOG_ITRawStatus](#) ()
Чтение немаскированного флага прерывания сторожевого таймера.
- [FlagStatus WATCHDOG_ITMaskedStatus](#) ()
Чтение маскированного флага прерывания сторожевого таймера.
- void [WATCHDOG_ITStatusClear](#) ()
Очищение статусного бита прерывания сторожевого таймера.

8.26.1 Подробное описание

Файл содержит все прототипы функций для сторожевого таймера.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

15.01.2016

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2016 ОАО "НИИЭТ"

8.26.2 Функции

8.26.2.1 void [WATCHDOG_Cmd](#) ([FunctionalState](#) State)

Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.26.2.2 uint32_t WATCHDOG_GetCounter ()

Получение текущего значения счетчика.

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4_watchdog.c строка 105

8.26.2.3 uint32_t WATCHDOG_GetReload ()

Получение текущего значения перезагрузки.

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4_watchdog.c строка 95

8.26.2.4 FlagStatus WATCHDOG_ITMaskedStatus ()

Чтение маскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 174

8.26.2.5 FlagStatus WATCHDOG_ITRawStatus ()

Чтение немаскированного флага прерывания сторожевого таймера.

Возвращаемые значения

Status	Статус прерывания.
--------	--------------------

См. определение в файле niietcm4_watchdog.c строка 153

8.26.2.6 void WATCHDOG_ITStatusClear ()

Очищение статусного бита прерывания сторожевого таймера.

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 195

8.26.2.7 void WATCHDOG_LockCmd (FunctionalState State)

Запрещение записи во все регистры сторожевого таймера для предотвращения отключения его сбойными программами.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 134

Перекрестные ссылки IS_FUNCTIONAL_STATE, WATCHDOG_Lock_Value и WATCHDOG_Unlock_Value.

8.26.2.8 void WATCHDOG_RstCmd (FunctionalState State)

Разрешение сброса по сторожевому таймеру. Сброс будет произведен когда счетчик досчитает до нуля при установленном ранее и несброшенном флаге прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState .
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 119

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.26.2.9 void WATCHDOG_SetReload (uint32_t ReloadVal)

Установка значения перезагрузки.

Аргументы

ReloadVal	Значение перезагрузки. Параметр принимает любое значение из диапазона 0x1 - 0xFFFFFFFF.
-----------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_watchdog.c строка 83

Перекрестные ссылки IS_WATCHDOG_RELOAD.

Предметный указатель

- `_CHANNEL_CFG_bits`, 369
 - `CYCLE_CTRL`, 369
 - `DST_INC`, 369
 - `DST_PROT_BUFFERABLE`, 370
 - `DST_PROT_CACHEABLE`, 370
 - `DST_PROT_PRIVILEGED`, 370
 - `DST_SIZE`, 370
 - `N_MINUS_1`, 370
 - `NEXT_USEBURST`, 370
 - `R_POWER`, 370
 - `SRC_INC`, 371
 - `SRC_PROT_BUFFERABLE`, 371
 - `SRC_PROT_CACHEABLE`, 371
 - `SRC_PROT_PRIVILEGED`, 371
 - `SRC_SIZE`, 371
- Битовые операции, 156
 - `GPIO_ClearBits`, 156
 - `GPIO_SetBits`, 156
 - `GPIO_ToggleBits`, 156
- Цифровые компараторы, 46, 55
 - `ADC_DC_DeInit`, 46
 - `ADC_DC_ITCmd`, 55
 - `ADC_DC_ITConfig`, 55
 - `ADC_DC_ITGenCmd`, 57
 - `ADC_DC_ITMaskCmd`, 57
 - `ADC_DC_ITMaskedStatus`, 57
 - `ADC_DC_ITRawStatus`, 58
 - `ADC_DC_ITStatusClear`, 58
 - `ADC_DC_Init`, 46
 - `ADC_DC_StructInit`, 46
- Чтение, 152
 - `GPIO_Read`, 152
 - `GPIO_ReadBit`, 152
 - `GPIO_ReadMask`, 152
- Чтение и запись, 151
- Драйвер периферии, 362
- Фильтрация, 158
 - `GPIO_QualCmd`, 158
 - `GPIO_QualConfig`, 158
 - `GPIO_SyncCmd`, 159
- Функции, 37, 66, 78, 116, 134, 148, 171, 188, 191, 205, 222, 230
 - `ADC_Cmd`, 37
 - `ADC_DC_Cmd`, 38
 - `ADC_DC_GetLastData`, 38
 - `ADC_DC_TrigStatus`, 38
 - `ADC_DC_TrigStatusClear`, 39
 - `ADC_SEQ_Cmd`, 39
 - `ADC_SEQ_FIFOEmptyStatus`, 39
 - `ADC_SEQ_FIFOEmptyStatusClear`, 40
 - `ADC_SEQ_FIFOFullStatus`, 40
 - `ADC_SEQ_FIFOFullStatusClear`, 40
 - `ADC_SEQ_GetConversionCount`, 41
 - `ADC_SEQ_GetFIFOData`, 41
 - `ADC_SEQ_GetFIFOLoad`, 41
 - `ADC_SEQ_SWReq`, 41
 - `BOOTFLASH_ITCmd`, 66
 - `BOOTFLASH_Init`, 66
 - `BOOTFLASH_OperationStatus`, 66
 - `BOOTFLASH_OperationStatusClear`, 67
 - `EXTMEM_DeInit`, 134
 - `EXTMEM_Init`, 134
 - `EXTMEM_StructInit`, 134
 - `RCC_SysClkDiv2Out`, 171
 - `UART_BaudRateDivConfig`, 205
 - `UART_Break`, 205
 - `UART_Cmd`, 206
 - `USERFLASH_ITCmd`, 222
 - `USERFLASH_Init`, 222
 - `USERFLASH_OperationStatus`, 222
 - `USERFLASH_OperationStatusClear`, 223
- Инициализация, 43
- Информационная область флеш, 64, 70, 220, 226
 - `BOOTFLASH_INFO_PAGE_SIZE_BYTE←ES`, 64
 - `BOOTFLASH_INFO_PAGE_TOTAL`, 64
 - `BOOTFLASH_INFO_TOTAL_BYTES`, 64
 - `BOOTFLASH_Info_PageErase`, 70
 - `BOOTFLASH_Info_Write`, 70
 - `USERFLASH_INFO_PAGE_SIZE_BYTE←ES`, 220
 - `USERFLASH_INFO_PAGE_TOTAL`, 220
 - `USERFLASH_INFO_TOTAL_BYTES`, 220
 - `USERFLASH_Info_PageErase`, 226
 - `USERFLASH_Info_Read`, 226
 - `USERFLASH_Info_Write`, 226
- Инициализация и деинициализация, 149, 207
 - `GPIO_DeInit`, 149
 - `GPIO_Init`, 149
 - `GPIO_StructInit`, 150
 - `UART_DeInit`, 207
 - `UART_Init`, 207
 - `UART_StructInit`, 207
- Инициализация каналов DMA, 117
 - `DMA_ChannelDeInit`, 117
 - `DMA_ChannelInit`, 117
 - `DMA_ChannelStructInit`, 118
- Инициализация контроллера DMA, 119

- DMA_DeInit, 119
- DMA_Init, 119
- DMA_StructInit, 119
- Конфигурация, 79, 192, 231
 - CAP_DeInit, 79
 - CAP_GetShadowTimer, 79
 - CAP_GetTimer, 80
 - CAP_Init, 80
 - CAP_SetShadowTimer, 80
 - CAP_SetTimer, 80
 - CAP_StructInit, 82
 - CAP_SwSync, 82
 - CAP_SyncCmd, 82
 - CAP_TimerCmd, 82
 - TIMER_Cmd, 192
 - TIMER_ExtInputConfig, 192
 - TIMER_FreqConfig, 193
 - TIMER_GetCounter, 193
 - TIMER_GetReload, 193
 - TIMER_PeriodConfig, 194
 - TIMER_SetCounter, 194
 - TIMER_SetReload, 194
 - WATCHDOG_Cmd, 231
 - WATCHDOG_GetCounter, 231
 - WATCHDOG_GetReload, 231
 - WATCHDOG_LockCmd, 232
 - WATCHDOG_RstCmd, 232
 - WATCHDOG_SetReload, 232
- Конфигурация PLL, 172
 - RCC_PLLAutoConfig, 172
 - RCC_PLLDeInit, 172
 - RCC_PLLInit, 173
 - RCC_PLLPowerDownCmd, 173
 - RCC_PLLStructInit, 174
- Конфигурация контроллера DMA, 121
 - DMA_BasePtrConfig, 121
 - DMA_ChannelEnableCmd, 121
 - DMA_HighPriorityCmd, 122
 - DMA_MasterEnableCmd, 122
 - DMA_PrmAltCmd, 122
 - DMA_ProtectionConfig, 123
 - DMA_ReqMaskCmd, 123
 - DMA_SWRequestCmd, 123
 - DMA_UseBurstCmd, 124
- Конфигурация прерываний, 54
- Конфигурация секвенсоров для DMA, 51
 - ADC_SEQ_DMACmd, 51
 - ADC_SEQ_DMAConfig, 51
 - ADC_SEQ_DMAErrorStatus, 51
 - ADC_SEQ_DMAErrorStatusClear, 53
- Константы, 15, 62, 75, 97, 127, 143, 162, 190, 204, 218, 229
 - RCC_CLK_CHANGE_TIMEOUT, 162
 - RCC_CLK_PLL_STABLE_TIMEOUT, 162
- Макросы, 364
- Маски адреса, 128
 - EXTMEM_CEMask_Addr_11_19, 128
- EXTMEM_CEMask_Addr_12, 128
- EXTMEM_CEMask_Addr_13, 128
- EXTMEM_CEMask_Addr_14, 128
- EXTMEM_CEMask_Addr_15, 128
- EXTMEM_CEMask_Addr_16, 128
- EXTMEM_CEMask_Addr_17, 129
- EXTMEM_CEMask_Addr_18, 129
- EXTMEM_CEMask_Addr_19, 129
- Маски для CHANNEL_CFG, 98
 - CHANNEL_CFG_CYCLE_CTRL_Msk, 98
 - CHANNEL_CFG_CYCLE_CTRL_Pos, 98
 - CHANNEL_CFG_DST_INC_Msk, 98
 - CHANNEL_CFG_DST_INC_Pos, 98
 - CHANNEL_CFG_DST_PROT_CTRL_← Msk, 98
 - CHANNEL_CFG_DST_PROT_CTRL_← Pos, 99
 - CHANNEL_CFG_DST_SIZE_Msk, 99
 - CHANNEL_CFG_DST_SIZE_Pos, 99
 - CHANNEL_CFG_N_MINUS_1_Msk, 99
 - CHANNEL_CFG_N_MINUS_1_Pos, 99
 - CHANNEL_CFG_NEXT_USEBURST_← Msk, 99
 - CHANNEL_CFG_NEXT_USEBURST_← Pos, 99
 - CHANNEL_CFG_R_POWER_Msk, 99
 - CHANNEL_CFG_R_POWER_Pos, 99
 - CHANNEL_CFG_SRC_INC_Msk, 100
 - CHANNEL_CFG_SRC_INC_Pos, 100
 - CHANNEL_CFG_SRC_PROT_CTRL_← Msk, 100
 - CHANNEL_CFG_SRC_PROT_CTRL_← Pos, 100
 - CHANNEL_CFG_SRC_SIZE_Msk, 100
 - CHANNEL_CFG_SRC_SIZE_Pos, 100
- Маски источников прерываний, 76
 - CAP_ITSource_All, 76
 - CAP_ITSource_CapEvent0, 76
 - CAP_ITSource_CapEvent1, 76
 - CAP_ITSource_CapEvent2, 76
 - CAP_ITSource_CapEvent3, 76
 - CAP_ITSource_GeneralInt, 76
 - CAP_ITSource_TimerEqCompare, 77
 - CAP_ITSource_TimerEqPeriod, 77
 - CAP_ITSource_TimerOvf, 77
 - IS_CAP_IT_SOURCE_SINGLE, 77
- Маски каналов DMA, 101
 - DMA_Channel_All, 101
 - IS_GET_DMA_CHANNEL, 101
- Маски каналов для измерений, 16
 - ADC_Channel_0, 16
 - ADC_Channel_1, 16
 - ADC_Channel_10, 16
 - ADC_Channel_11, 16
 - ADC_Channel_12, 17
 - ADC_Channel_13, 17
 - ADC_Channel_14, 17

- ADC_Channel_15, [17](#)
- ADC_Channel_16, [17](#)
- ADC_Channel_17, [17](#)
- ADC_Channel_18, [17](#)
- ADC_Channel_19, [17](#)
- ADC_Channel_2, [17](#)
- ADC_Channel_20, [18](#)
- ADC_Channel_21, [18](#)
- ADC_Channel_22, [18](#)
- ADC_Channel_23, [18](#)
- ADC_Channel_3, [18](#)
- ADC_Channel_4, [18](#)
- ADC_Channel_5, [18](#)
- ADC_Channel_6, [18](#)
- ADC_Channel_7, [18](#)
- ADC_Channel_8, [18](#)
- ADC_Channel_9, [19](#)
- ADC_Channel_All, [19](#)
- ADC_Channel_None, [19](#)
- Маски каналов по имени, [103](#)
 - DMA_Channel_ADCSEQ0, [103](#)
 - DMA_Channel_ADCSEQ1, [103](#)
 - DMA_Channel_ADCSEQ2, [103](#)
 - DMA_Channel_ADCSEQ3, [103](#)
 - DMA_Channel_ADCSEQ4, [103](#)
 - DMA_Channel_ADCSEQ5, [104](#)
 - DMA_Channel_ADCSEQ6, [104](#)
 - DMA_Channel_ADCSEQ7, [104](#)
 - DMA_Channel_SPI0_RX, [104](#)
 - DMA_Channel_SPI0_TX, [104](#)
 - DMA_Channel_SPI1_RX, [104](#)
 - DMA_Channel_SPI1_TX, [104](#)
 - DMA_Channel_SPI2_RX, [104](#)
 - DMA_Channel_SPI2_TX, [104](#)
 - DMA_Channel_SPI3_RX, [105](#)
 - DMA_Channel_SPI3_TX, [105](#)
 - DMA_Channel_UART0_RX, [105](#)
 - DMA_Channel_UART0_TX, [105](#)
 - DMA_Channel_UART1_RX, [105](#)
 - DMA_Channel_UART1_TX, [105](#)
 - DMA_Channel_UART2_RX, [105](#)
 - DMA_Channel_UART2_TX, [105](#)
 - DMA_Channel_UART3_RX, [105](#)
 - DMA_Channel_UART3_TX, [105](#)
- Маски каналов по номеру, [107](#)
 - DMA_Channel_0, [107](#)
 - DMA_Channel_1, [107](#)
 - DMA_Channel_10, [107](#)
 - DMA_Channel_11, [107](#)
 - DMA_Channel_12, [107](#)
 - DMA_Channel_13, [108](#)
 - DMA_Channel_14, [108](#)
 - DMA_Channel_15, [108](#)
 - DMA_Channel_16, [108](#)
 - DMA_Channel_17, [108](#)
 - DMA_Channel_18, [108](#)
 - DMA_Channel_19, [108](#)
 - DMA_Channel_2, [108](#)
 - DMA_Channel_20, [108](#)
 - DMA_Channel_21, [109](#)
 - DMA_Channel_22, [109](#)
 - DMA_Channel_23, [109](#)
 - DMA_Channel_3, [109](#)
 - DMA_Channel_4, [109](#)
 - DMA_Channel_5, [109](#)
 - DMA_Channel_6, [109](#)
 - DMA_Channel_7, [109](#)
 - DMA_Channel_8, [109](#)
 - DMA_Channel_9, [109](#)
- Маски пинов, [144](#)
 - GPIO_Pin_0, [144](#)
 - GPIO_Pin_0_3, [144](#)
 - GPIO_Pin_0_7, [144](#)
 - GPIO_Pin_1, [144](#)
 - GPIO_Pin_10, [145](#)
 - GPIO_Pin_11, [145](#)
 - GPIO_Pin_12, [145](#)
 - GPIO_Pin_12_15, [145](#)
 - GPIO_Pin_13, [145](#)
 - GPIO_Pin_14, [145](#)
 - GPIO_Pin_15, [145](#)
 - GPIO_Pin_2, [145](#)
 - GPIO_Pin_3, [145](#)
 - GPIO_Pin_4, [146](#)
 - GPIO_Pin_4_7, [146](#)
 - GPIO_Pin_5, [146](#)
 - GPIO_Pin_6, [146](#)
 - GPIO_Pin_7, [146](#)
 - GPIO_Pin_8, [146](#)
 - GPIO_Pin_8_11, [146](#)
 - GPIO_Pin_8_15, [146](#)
 - GPIO_Pin_9, [146](#)
 - GPIO_Pin_All, [146](#)
 - IS_GET_GPIO_PIN, [147](#)
- Маски портов, [303](#)
 - GPIO_Regs_A_C_E_G_Mask, [303](#)
 - GPIO_Regs_B_D_F_H_Mask, [303](#)
 - GPIO_Regs_GPIOA_Mask, [303](#)
 - GPIO_Regs_GPIOB_Mask, [303](#)
 - GPIO_Regs_GPIOC_Mask, [303](#)
 - GPIO_Regs_GPIOD_Mask, [303](#)
 - GPIO_Regs_GPIOE_Mask, [303](#)
 - GPIO_Regs_GPIOF_Mask, [304](#)
 - GPIO_Regs_GPIOG_Mask, [304](#)
 - GPIO_Regs_GPIOH_Mask, [304](#)
- Маски выбора цифровых компараторов, [20](#)
 - ADC_DC_0, [20](#)
 - ADC_DC_1, [20](#)
 - ADC_DC_10, [20](#)
 - ADC_DC_11, [20](#)
 - ADC_DC_12, [21](#)
 - ADC_DC_13, [21](#)
 - ADC_DC_14, [21](#)
 - ADC_DC_15, [21](#)
 - ADC_DC_16, [21](#)
 - ADC_DC_17, [21](#)

- ADC_DC_18, 21
- ADC_DC_19, 21
- ADC_DC_2, 21
- ADC_DC_20, 22
- ADC_DC_21, 22
- ADC_DC_22, 22
- ADC_DC_23, 22
- ADC_DC_3, 22
- ADC_DC_4, 22
- ADC_DC_5, 22
- ADC_DC_6, 22
- ADC_DC_7, 22
- ADC_DC_8, 22
- ADC_DC_9, 23
- ADC_DC_All, 23
- ADC_DC_None, 23
- Маски выбора секвенсоров, 24
 - ADC_SEQ_0, 24
 - ADC_SEQ_1, 24
 - ADC_SEQ_2, 24
 - ADC_SEQ_3, 24
 - ADC_SEQ_4, 24
 - ADC_SEQ_5, 24
 - ADC_SEQ_6, 24
 - ADC_SEQ_7, 25
- Модули АЦП, 44
 - ADC_DeInit, 44
 - ADC_Init, 44
 - ADC_StructInit, 44
- Начальные значения регистров, 237, 282, 294, 300, 316
 - EXT_MEM_CFG_Reset_Value, 294
 - GPIO_DATAOUT_Reset_Value, 300
 - GPIO_GPIODEN0_Reset_Value, 300
 - GPIO_GPIODEN1_Reset_Value, 300
 - GPIO_GPIODEN2_Reset_Value, 300
 - GPIO_GPIODEN3_Reset_Value, 300
 - GPIO_GPIOODCTLx_Reset_Value, 301
 - GPIO_GPIOODSCTLx_Reset_Value, 301
 - GPIO_GPIOPCTLx_Reset_Value, 301
 - GPIO_GPIOPUCTLx_Reset_Value, 301
 - GPIO_GPIOQEx_Reset_Value, 301
 - GPIO_GPIOQMx_Reset_Value, 301
 - GPIO_GPIOQPx_Reset_Value, 301
 - GPIO_GPIOSEx_Reset_Value, 301
 - RCC_PLL_CTRL_Reset_Value, 316
 - RCC_PLL_NF_Reset_Value, 316
 - RCC_PLL_NR_Reset_Value, 316
 - RCC_PLL_OD_Reset_Value, 316
- Настройка DMA, 217
 - UART_DMABlkOnErrCmd, 217
 - UART_DMACmd, 217
- Настройка драйвера, 363
 - EXT_OSC_VALUE, 363
 - INT_OSC_VALUE, 363
- Основная область флеш, 63, 68, 219, 224
 - BOOTFLASH_FullErase, 68
 - BOOTFLASH_PAGE_SIZE_BYTES, 63
 - BOOTFLASH_PAGE_TOTAL, 63
 - BOOTFLASH_PageErase, 68
 - BOOTFLASH_TOTAL_BYTES, 63
 - BOOTFLASH_Write, 68
 - USERFLASH_FullErase, 224
 - USERFLASH_PAGE_SIZE_BYTES, 219
 - USERFLASH_PAGE_TOTAL, 219
 - USERFLASH_PageErase, 224
 - USERFLASH_Read, 224
 - USERFLASH_TOTAL_BYTES, 219
 - USERFLASH_Write, 225
- Периферия, 368
- Прерывания, 94, 160, 196, 214, 233
 - CAP_ITCmd, 94
 - CAP_ITForceCmd, 94
 - CAP_ITPendClear, 95
 - CAP_ITPendStatus, 95
 - CAP_ITStatus, 95
 - CAP_ITStatusClear, 95
 - GPIO_ITCmd, 160
 - GPIO_ITConfig, 160
 - GPIO_ITStatusClear, 160
 - TIMER_ITCmd, 196
 - TIMER_ITStatus, 196
 - TIMER_ITStatusClear, 196
 - UART_ITCmd, 214
 - UART_ITFIFOLevelConfig, 214
 - UART_ITMaskedStatus, 215
 - UART_ITRawStatus, 215
 - UART_ITStatusClear, 215
 - WATCHDOG_ITMaskedStatus, 233
 - WATCHDOG_ITRawStatus, 233
 - WATCHDOG_ITStatusClear, 233
- Прием и передача, 210
 - UART_ErrorStatus, 210
 - UART_ErrorStatusClear, 210
 - UART_FlagStatus, 210
 - UART_RecieveData, 212
 - UART_SendData, 212
- Приватные данные, 235, 256, 262, 280, 292, 298, 314, 327, 330, 337, 350, 357
- Приватные функции, 238, 258, 264, 283, 295, 305, 317, 328, 332, 339, 352, 359
 - ADC_Cmd, 240
 - ADC_DC_Cmd, 240
 - ADC_DC_DeInit, 240
 - ADC_DC_GetLastData, 240
 - ADC_DC_ITCmd, 241
 - ADC_DC_ITConfig, 241
 - ADC_DC_ITGenCmd, 242
 - ADC_DC_ITMaskCmd, 242
 - ADC_DC_ITMaskedStatus, 242
 - ADC_DC_ITRawStatus, 243
 - ADC_DC_ITStatusClear, 243
 - ADC_DC_Init, 241
 - ADC_DC_StructInit, 243
 - ADC_DC_TrigStatus, 244
 - ADC_DC_TrigStatusClear, 244

- ADC_DeInit, 244
- ADC_Init, 244
- ADC_SEQ_Cmd, 246
- ADC_SEQ_DMACmd, 246
- ADC_SEQ_DMAConfig, 247
- ADC_SEQ_DMAErrorStatus, 247
- ADC_SEQ_DMAErrorStatusClear, 247
- ADC_SEQ_DeInit, 246
- ADC_SEQ_FIFOEmptyStatus, 248
- ADC_SEQ_FIFOEmptyStatusClear, 248
- ADC_SEQ_FIFOFullStatus, 248
- ADC_SEQ_FIFOFullStatusClear, 249
- ADC_SEQ_GetConversionCount, 249
- ADC_SEQ_GetFIFOData, 249
- ADC_SEQ_GetFIFOLoad, 249
- ADC_SEQ_GetITCount, 251
- ADC_SEQ_ITCmd, 251
- ADC_SEQ_ITConfig, 252
- ADC_SEQ_ITCountRst, 252
- ADC_SEQ_ITMaskedStatus, 252
- ADC_SEQ_ITRawStatus, 253
- ADC_SEQ_ITStatusClear, 253
- ADC_SEQ_Init, 251
- ADC_SEQ_SWReq, 254
- ADC_SEQ_StructInit, 253
- ADC_StructInit, 254
- BOOTFLASH_FullErase, 258
- BOOTFLASH_ITCmd, 259
- BOOTFLASH_Info_PageErase, 258
- BOOTFLASH_Info_Write, 259
- BOOTFLASH_Init, 259
- BOOTFLASH_OperationStatus, 259
- BOOTFLASH_OperationStatusClear, 260
- BOOTFLASH_PageErase, 260
- BOOTFLASH_Write, 260
- CAP_Capture_Cmd, 265
- CAP_Capture_GetCap0, 265
- CAP_Capture_GetCap1, 266
- CAP_Capture_GetCap2, 266
- CAP_Capture_GetCap3, 266
- CAP_Capture_Init, 266
- CAP_Capture_SetCap0, 268
- CAP_Capture_SetCap1, 268
- CAP_Capture_SetCap2, 268
- CAP_Capture_SetCap3, 269
- CAP_Capture_StructInit, 269
- CAP_DeInit, 269
- CAP_GetShadowTimer, 270
- CAP_GetTimer, 270
- CAP_ITCmd, 270
- CAP_ITForceCmd, 271
- CAP_ITPendClear, 271
- CAP_ITPendStatus, 271
- CAP_ITStatus, 272
- CAP_ITStatusClear, 272
- CAP_Init, 270
- CAP_PWM_GetCompare, 272
- CAP_PWM_GetPeriod, 272
- CAP_PWM_GetShadowCompare, 274
- CAP_PWM_GetShadowPeriod, 274
- CAP_PWM_Init, 274
- CAP_PWM_SetCompare, 274
- CAP_PWM_SetPeriod, 275
- CAP_PWM_SetShadowCompare, 275
- CAP_PWM_SetShadowPeriod, 275
- CAP_PWM_StructInit, 276
- CAP_SetShadowTimer, 276
- CAP_SetTimer, 276
- CAP_StructInit, 277
- CAP_SwSync, 277
- CAP_SyncCmd, 277
- CAP_TimerCmd, 277
- DMA_BasePtrConfig, 283
- DMA_ChannelDeInit, 284
- DMA_ChannelEnableCmd, 284
- DMA_ChannelInit, 284
- DMA_ChannelStructInit, 285
- DMA_ClearErrorStatus, 285
- DMA_DeInit, 285
- DMA_ErrorStatus, 287
- DMA_HighPriorityCmd, 287
- DMA_Init, 287
- DMA_MasterEnableCmd, 288
- DMA_MasterEnableStatus, 288
- DMA_PrmAltCmd, 288
- DMA_ProtectionConfig, 288
- DMA_ReqMaskCmd, 289
- DMA_SWRequestCmd, 290
- DMA_StateStatus, 289
- DMA_StructInit, 289
- DMA_UseBurstCmd, 290
- DMA_WaitOnReqStatus, 290
- EXTMEM_DeInit, 295
- EXTMEM_Init, 295
- EXTMEM_StructInit, 295
- GPIO_ClearBits, 305
- GPIO_DeInit, 306
- GPIO_ITCmd, 306
- GPIO_ITConfig, 308
- GPIO_ITStatusClear, 308
- GPIO_Init, 306
- GPIO_QualCmd, 308
- GPIO_QualConfig, 309
- GPIO_Read, 309
- GPIO_ReadBit, 309
- GPIO_ReadMask, 310
- GPIO_SetBits, 310
- GPIO_StructInit, 310
- GPIO_SyncCmd, 311
- GPIO_ToggleBits, 311
- GPIO_Write, 311
- GPIO_WriteBit, 312
- GPIO_WriteMask, 312
- RCC_ADCClkCmd, 318
- RCC_ADCClkDivConfig, 318
- RCC_PLLAutoConfig, 319

- RCC_PLLDeInit, 320
- RCC_PLLInit, 320
- RCC_PLLPowerDownCmd, 321
- RCC_PLLStructInit, 321
- RCC_PeriphClkCmd, 318
- RCC_PeriphRstCmd, 319
- RCC_SPIClkCmd, 321
- RCC_SPIClkDivConfig, 322
- RCC_SPIClkSel, 322
- RCC_SysClkDiv2Out, 322
- RCC_SysClkSel, 323
- RCC_SysClkStatus, 323
- RCC_UARTClkCmd, 323
- RCC_UARTClkDivConfig, 324
- RCC_UARTClkSel, 324
- RCC_USBClkCmd, 324
- RCC_USBClkConfig, 325
- RCC_WaitClkChange, 325
- TIMER_Cmd, 332
- TIMER_ExtInputConfig, 332
- TIMER_FreqConfig, 333
- TIMER_GetCounter, 333
- TIMER_GetReload, 333
- TIMER_ITCmd, 334
- TIMER_ITStatus, 334
- TIMER_ITStatusClear, 334
- TIMER_PeriodConfig, 334
- TIMER_SetCounter, 335
- TIMER_SetReload, 335
- UART_BaudRateDivConfig, 340
- UART_Break, 340
- UART_Cmd, 340
- UART_DMABlkOnErrCmd, 341
- UART_DMACmd, 341
- UART_DeInit, 341
- UART_ErrorStatus, 341
- UART_ErrorStatusClear, 342
- UART_FlagStatus, 342
- UART_ITCmd, 343
- UART_ITFIFOLevelConfig, 343
- UART_ITMaskedStatus, 343
- UART_ITRawStatus, 345
- UART_ITStatusClear, 345
- UART_Init, 342
- UART_ModemConfig, 345
- UART_ModemStructInit, 346
- UART_RecieveData, 346
- UART_SendData, 346
- UART_StructInit, 346
- USERFLASH_FullErase, 352
- USERFLASH_ITCmd, 353
- USERFLASH_Info_PageErase, 352
- USERFLASH_Info_Read, 353
- USERFLASH_Info_Write, 353
- USERFLASH_Init, 353
- USERFLASH_OperationStatus, 354
- USERFLASH_OperationStatusClear, 354
- USERFLASH_PageErase, 354
- USERFLASH_Read, 354
- USERFLASH_Write, 355
- WATCHDOG_Cmd, 359
- WATCHDOG_GetCounter, 359
- WATCHDOG_GetReload, 359
- WATCHDOG_ITMaskedStatus, 360
- WATCHDOG_ITRawStatus, 360
- WATCHDOG_ITStatusClear, 360
- WATCHDOG_LockCmd, 360
- WATCHDOG_RstCmd, 360
- WATCHDOG_SetReload, 361
- Приватные константы, 236, 257, 263, 281, 293, 299, 315, 331, 338, 351, 358
- WATCHDOG_Lock_Value, 358
- WATCHDOG_Unlock_Value, 358
- Режим ШИМ, 84
 - CAP_PWM_GetCompare, 84
 - CAP_PWM_GetPeriod, 84
 - CAP_PWM_GetShadowCompare, 85
 - CAP_PWM_GetShadowPeriod, 85
 - CAP_PWM_Init, 85
 - CAP_PWM_SetCompare, 85
 - CAP_PWM_SetPeriod, 87
 - CAP_PWM_SetShadowCompare, 87
 - CAP_PWM_SetShadowPeriod, 87
 - CAP_PWM_StructInit, 88
- Режим модема, 213
 - UART_ModemConfig, 213
 - UART_ModemStructInit, 213
- Режим захвата, 89
 - CAP_Capture_Cmd, 89
 - CAP_Capture_GetCap0, 89
 - CAP_Capture_GetCap1, 90
 - CAP_Capture_GetCap2, 90
 - CAP_Capture_GetCap3, 90
 - CAP_Capture_Init, 90
 - CAP_Capture_SetCap0, 92
 - CAP_Capture_SetCap1, 92
 - CAP_Capture_SetCap2, 92
 - CAP_Capture_SetCap3, 93
 - CAP_Capture_StructInit, 93
- Секвенсоры, 48, 59
 - ADC_SEQ_DeInit, 48
 - ADC_SEQ_GetITCount, 59
 - ADC_SEQ_ITCmd, 59
 - ADC_SEQ_ITConfig, 60
 - ADC_SEQ_ITCountRst, 60
 - ADC_SEQ_ITMaskedStatus, 60
 - ADC_SEQ_ITRawStatus, 61
 - ADC_SEQ_ITStatusClear, 61
 - ADC_SEQ_Init, 48
 - ADC_SEQ_StructInit, 48
- Статусная информация, 125
 - DMA_ClearErrorStatus, 125
 - DMA_ErrorStatus, 125
 - DMA_MasterEnableStatus, 125
 - DMA_StateStatus, 125
 - DMA_WaitOnReqStatus, 126

- Тактирование ADC, [183](#)
 - RCC_ADCClkCmd, [183](#)
 - RCC_ADCClkDivConfig, [183](#)
- Тактирование SPI, [181](#)
 - RCC_SPIClkCmd, [181](#)
 - RCC_SPIClkDivConfig, [181](#)
 - RCC_SPIClkSel, [181](#)
- Тактирование UART, [178](#)
 - RCC_UARTClkCmd, [178](#)
 - RCC_UARTClkDivConfig, [178](#)
 - RCC_UARTClkSel, [178](#)
- Тактирование USB, [177](#)
 - RCC_USBClkCmd, [177](#)
 - RCC_USBClkConfig, [177](#)
- Типы, [26](#), [65](#), [71](#), [111](#), [130](#), [136](#), [163](#), [185](#), [189](#), [198](#), [221](#), [228](#), [365](#)
 - ADC_Average_16, [32](#)
 - ADC_Average_2, [32](#)
 - ADC_Average_32, [32](#)
 - ADC_Average_4, [32](#)
 - ADC_Average_64, [32](#)
 - ADC_Average_8, [32](#)
 - ADC_Average_Disable, [32](#)
 - ADC_Average_TypeDef, [32](#)
 - ADC_DC_Channel_0, [32](#)
 - ADC_DC_Channel_1, [32](#)
 - ADC_DC_Channel_10, [32](#)
 - ADC_DC_Channel_11, [32](#)
 - ADC_DC_Channel_12, [32](#)
 - ADC_DC_Channel_13, [32](#)
 - ADC_DC_Channel_14, [32](#)
 - ADC_DC_Channel_15, [33](#)
 - ADC_DC_Channel_16, [33](#)
 - ADC_DC_Channel_17, [33](#)
 - ADC_DC_Channel_18, [33](#)
 - ADC_DC_Channel_19, [33](#)
 - ADC_DC_Channel_2, [32](#)
 - ADC_DC_Channel_20, [33](#)
 - ADC_DC_Channel_21, [33](#)
 - ADC_DC_Channel_22, [33](#)
 - ADC_DC_Channel_23, [33](#)
 - ADC_DC_Channel_3, [32](#)
 - ADC_DC_Channel_4, [32](#)
 - ADC_DC_Channel_5, [32](#)
 - ADC_DC_Channel_6, [32](#)
 - ADC_DC_Channel_7, [32](#)
 - ADC_DC_Channel_8, [32](#)
 - ADC_DC_Channel_9, [32](#)
 - ADC_DC_Channel_None, [33](#)
 - ADC_DC_Channel_TypeDef, [32](#)
 - ADC_DC_Condition_High, [33](#)
 - ADC_DC_Condition_Low, [33](#)
 - ADC_DC_Condition_TypeDef, [33](#)
 - ADC_DC_Condition_Window, [33](#)
 - ADC_DC_Mode_Multiple, [33](#)
 - ADC_DC_Mode_MultipleHyst, [33](#)
 - ADC_DC_Mode_Single, [33](#)
 - ADC_DC_Mode_SingleHyst, [33](#)
 - ADC_DC_Mode_TypeDef, [33](#)
 - ADC_DC_Module_0, [33](#)
 - ADC_DC_Module_1, [33](#)
 - ADC_DC_Module_10, [34](#)
 - ADC_DC_Module_11, [34](#)
 - ADC_DC_Module_12, [34](#)
 - ADC_DC_Module_13, [34](#)
 - ADC_DC_Module_14, [34](#)
 - ADC_DC_Module_15, [34](#)
 - ADC_DC_Module_16, [34](#)
 - ADC_DC_Module_17, [34](#)
 - ADC_DC_Module_18, [34](#)
 - ADC_DC_Module_19, [34](#)
 - ADC_DC_Module_2, [33](#)
 - ADC_DC_Module_20, [34](#)
 - ADC_DC_Module_21, [34](#)
 - ADC_DC_Module_22, [34](#)
 - ADC_DC_Module_23, [34](#)
 - ADC_DC_Module_3, [33](#)
 - ADC_DC_Module_4, [33](#)
 - ADC_DC_Module_5, [33](#)
 - ADC_DC_Module_6, [34](#)
 - ADC_DC_Module_7, [34](#)
 - ADC_DC_Module_8, [34](#)
 - ADC_DC_Module_9, [34](#)
 - ADC_DC_Module_TypeDef, [33](#)
 - ADC_Measure_Diff, [34](#)
 - ADC_Measure_Single, [34](#)
 - ADC_Measure_TypeDef, [34](#)
 - ADC_Mode_Active, [34](#)
 - ADC_Mode_Powerdown, [34](#)
 - ADC_Mode_StandBy, [34](#)
 - ADC_Mode_TypeDef, [34](#)
 - ADC_Module_0, [35](#)
 - ADC_Module_1, [35](#)
 - ADC_Module_10, [35](#)
 - ADC_Module_11, [35](#)
 - ADC_Module_2, [35](#)
 - ADC_Module_3, [35](#)
 - ADC_Module_4, [35](#)
 - ADC_Module_5, [35](#)
 - ADC_Module_6, [35](#)
 - ADC_Module_7, [35](#)
 - ADC_Module_8, [35](#)
 - ADC_Module_9, [35](#)
 - ADC_Module_TypeDef, [34](#)
 - ADC_Resolution_10bit, [35](#)
 - ADC_Resolution_12bit, [35](#)
 - ADC_Resolution_TypeDef, [35](#)
 - ADC_SEQ_FIFOLevel_1, [35](#)
 - ADC_SEQ_FIFOLevel_16, [35](#)
 - ADC_SEQ_FIFOLevel_2, [35](#)
 - ADC_SEQ_FIFOLevel_32, [35](#)
 - ADC_SEQ_FIFOLevel_4, [35](#)
 - ADC_SEQ_FIFOLevel_8, [35](#)
 - ADC_SEQ_FIFOLevel_TypeDef, [35](#)
 - ADC_SEQ_Module_0, [36](#)
 - ADC_SEQ_Module_1, [36](#)

- ADC_SEQ_Module_2, 36
- ADC_SEQ_Module_3, 36
- ADC_SEQ_Module_4, 36
- ADC_SEQ_Module_5, 36
- ADC_SEQ_Module_6, 36
- ADC_SEQ_Module_7, 36
- ADC_SEQ_Module_TypeDef, 35
- ADC_SEQ_StartEvent_CMP0, 36
- ADC_SEQ_StartEvent_CMP1, 36
- ADC_SEQ_StartEvent_CMP2, 36
- ADC_SEQ_StartEvent_Cycle, 36
- ADC_SEQ_StartEvent_ITGPIO, 36
- ADC_SEQ_StartEvent_PWM0, 36
- ADC_SEQ_StartEvent_PWM1, 36
- ADC_SEQ_StartEvent_PWM2, 36
- ADC_SEQ_StartEvent_PWM3, 36
- ADC_SEQ_StartEvent_PWM4, 36
- ADC_SEQ_StartEvent_PWM5, 36
- ADC_SEQ_StartEvent_SWReq, 36
- ADC_SEQ_StartEvent_TIM, 36
- ADC_SEQ_StartEvent_TypeDef, 36
- BOOTFLASH_Status_Complete, 65
- BOOTFLASH_Status_Error, 65
- BOOTFLASH_Status_None, 65
- BOOTFLASH_Status_TypeDef, 65
- Bit_CLEAR, 139
- Bit_SET, 139
- BitAction, 139
- CAP_Capture_Mode_Cycle, 73
- CAP_Capture_Mode_Single, 73
- CAP_Capture_Mode_TypeDef, 73
- CAP_Capture_Polarity_NegEdge, 73
- CAP_Capture_Polarity_PosEdge, 73
- CAP_Capture_Polarity_TypeDef, 73
- CAP_Halt_Free, 74
- CAP_Halt_Stop, 73
- CAP_Halt_StopOnZero, 73
- CAP_Halt_TypeDef, 73
- CAP_Mode_Capture, 74
- CAP_Mode_PWM, 74
- CAP_Mode_TypeDef, 74
- CAP_PWM_Polarity_Neg, 74
- CAP_PWM_Polarity_Pos, 74
- CAP_PWM_Polarity_TypeDef, 74
- CAP_SyncOut_Bypass, 74
- CAP_SyncOut_Disable, 74
- CAP_SyncOut_TimerEqPeriod, 74
- CAP_SyncOut_TypeDef, 74
- DMA_ArbitrationRate_1, 113
- DMA_ArbitrationRate_1024, 114
- DMA_ArbitrationRate_128, 114
- DMA_ArbitrationRate_16, 113
- DMA_ArbitrationRate_2, 113
- DMA_ArbitrationRate_256, 114
- DMA_ArbitrationRate_32, 114
- DMA_ArbitrationRate_4, 113
- DMA_ArbitrationRate_512, 114
- DMA_ArbitrationRate_64, 114
- DMA_ArbitrationRate_8, 113
- DMA_ArbitrationRate_TypeDef, 113
- DMA_DataInc_16, 114
- DMA_DataInc_32, 114
- DMA_DataInc_8, 114
- DMA_DataInc_Disable, 114
- DMA_DataInc_TypeDef, 114
- DMA_DataSize_16, 114
- DMA_DataSize_32, 114
- DMA_DataSize_8, 114
- DMA_DataSize_TypeDef, 114
- DMA_Mode_AltMemScatGath, 114
- DMA_Mode_AltPeriphScatGath, 114
- DMA_Mode_AutoReq, 114
- DMA_Mode_Basic, 114
- DMA_Mode_Disable, 114
- DMA_Mode_PingPong, 114
- DMA_Mode_PrmMemScatGath, 114
- DMA_Mode_PrmPeriphScatGath, 114
- DMA_Mode_TypeDef, 114
- DMA_State_Done, 115
- DMA_State_Free, 115
- DMA_State_Pause, 115
- DMA_State_PeriphScatGath, 115
- DMA_State_ReadConfigData, 115
- DMA_State_ReadDstDataEndPtr, 115
- DMA_State_ReadSrcData, 115
- DMA_State_ReadSrcDataEndPtr, 115
- DMA_State_TypeDef, 114
- DMA_State_WaitReq, 115
- DMA_State_WriteConfigData, 115
- DMA_State_WriteDstData, 115
- EXTMEM_RWWaitState_1, 132
- EXTMEM_RWWaitState_2, 132
- EXTMEM_RWWaitState_3, 132
- EXTMEM_RWWaitState_4, 132
- EXTMEM_RWWaitState_5, 132
- EXTMEM_RWWaitState_6, 132
- EXTMEM_RWWaitState_7, 132
- EXTMEM_RWWaitState_8, 132
- EXTMEM_RWWaitState_TypeDef, 132
- EXTMEM_ReadWaitState_1, 132
- EXTMEM_ReadWaitState_2, 132
- EXTMEM_ReadWaitState_3, 132
- EXTMEM_ReadWaitState_4, 132
- EXTMEM_ReadWaitState_5, 132
- EXTMEM_ReadWaitState_6, 132
- EXTMEM_ReadWaitState_7, 132
- EXTMEM_ReadWaitState_8, 132
- EXTMEM_ReadWaitState_TypeDef, 132
- EXTMEM_Width_16bit, 132
- EXTMEM_Width_8bit, 132
- EXTMEM_Width_TypeDef, 132
- EXTMEM_WriteWaitState_1, 133
- EXTMEM_WriteWaitState_2, 133
- EXTMEM_WriteWaitState_3, 133
- EXTMEM_WriteWaitState_4, 133
- EXTMEM_WriteWaitState_5, 133

- EXTMEM_WriteWaitState_6, 133
- EXTMEM_WriteWaitState_7, 133
- EXTMEM_WriteWaitState_8, 133
- EXTMEM_WriteWaitState_TypeDef, 132
- GPIO_AltFunc_1, 140
- GPIO_AltFunc_2, 140
- GPIO_AltFunc_3, 140
- GPIO_AltFunc_TypeDef, 139
- GPIO_Dir_In, 140
- GPIO_Dir_Out, 140
- GPIO_Dir_TypeDef, 140
- GPIO_IntPol_Neg, 140
- GPIO_IntPol_Pos, 140
- GPIO_IntPol_TypeDef, 140
- GPIO_IntType_Edge, 140
- GPIO_IntType_Level, 140
- GPIO_IntType_TypeDef, 140
- GPIO_Load_16mA, 140
- GPIO_Load_8mA, 140
- GPIO_Load_TypeDef, 140
- GPIO_Mode_AltFunc, 141
- GPIO_Mode_IO, 141
- GPIO_Mode_TypeDef, 140
- GPIO_Out_Dis, 141
- GPIO_Out_En, 141
- GPIO_Out_TypeDef, 141
- GPIO_OutMode_OD, 141
- GPIO_OutMode_PP, 141
- GPIO_OutMode_TypeDef, 141
- GPIO_PullUp_Dis, 141
- GPIO_PullUp_En, 141
- GPIO_PullUp_TypeDef, 141
- GPIO_Qual_Dis, 141
- GPIO_Qual_En, 141
- GPIO_Qual_TypeDef, 141
- GPIO_QualMode_3sample, 142
- GPIO_QualMode_6sample, 142
- GPIO_QualMode_TypeDef, 141
- GPIO_Sync_Dis, 142
- GPIO_Sync_En, 142
- GPIO_Sync_TypeDef, 142
- IS_ADC_AVERAGE, 28
- IS_ADC_DC_CHANNEL, 28
- IS_ADC_DC_CONDITION, 29
- IS_ADC_DC_MODE, 29
- IS_ADC_DC_MODULE, 29
- IS_ADC_MEASURE, 30
- IS_ADC_MODE, 30
- IS_ADC_MODULE, 30
- IS_ADC_RESOLUTION, 30
- IS_ADC_SEQ_FIFO_LEVEL, 31
- IS_ADC_SEQ_MODULE, 31
- IS_ADC_SEQ_START_EVENT, 31
- IS_BOOTFLASH_STATUS, 65
- IS_CAP_ALL_PERIPH, 365
- IS_CAP_CAPTURE_MODE, 72
- IS_CAP_CAPTURE_POLARITY, 72
- IS_CAP_HALT, 72
- IS_CAP_MODE, 72
- IS_CAP_PWM_POLARITY, 72
- IS_CAP_SYNC_OUT, 73
- IS_DMA_ARBITRATION_RATE, 112
- IS_DMA_DATA_INC, 112
- IS_DMA_DATA_SIZE, 112
- IS_DMA_MODE, 113
- IS_DMA_STATE, 113
- IS_EXTMEM_READ_WAITSTATE, 130
- IS_EXTMEM_RW_WAITSTATE, 131
- IS_EXTMEM_WIDTH, 131
- IS_EXTMEM_WRITE_WAITSTATE, 131
- IS_GPIO_ALL_PERIPH, 365
- IS_GPIO_ALT_FUNC, 137
- IS_GPIO_DIR, 137
- IS_GPIO_INT_POL, 137
- IS_GPIO_INT_TYPE, 137
- IS_GPIO_LOAD, 138
- IS_GPIO_MODE, 138
- IS_GPIO_OUT, 138
- IS_GPIO_OUT_MODE, 138
- IS_GPIO_PULLUP, 138
- IS_GPIO_QUAL, 139
- IS_GPIO_QUAL_MODE, 139
- IS_GPIO_SYNC, 139
- IS_RCC_ADC_CLK, 165
- IS_RCC_PERIPH_CLK, 165
- IS_RCC_PLL_NO, 165
- IS_RCC_PLL_REF, 165
- IS_RCC_SPI_CLK, 166
- IS_RCC_SYS_CLK, 166
- IS_RCC_UART_CLK, 166
- IS_RCC_USB_CLK, 166
- IS_RCC_USB_FREQ, 167
- IS_RTC_FORMAT, 186
- IS_RTC_MONTH, 186
- IS_RTC_WEEKDAY, 186
- IS_SPI_ALL_PERIPH, 366
- IS_TIMER_ALL_PERIPH, 366
- IS_TIMER_EXT_INPUT, 189
- IS_UART_ALL_PERIPH, 366
- IS_UART_DATA_WIDTH, 199
- IS_UART_DIR, 199
- IS_UART_ERROR, 199
- IS_UART_FIFO_LEVEL, 200
- IS_UART_FLAG, 200
- IS_UART_GET_IT_SOURCE, 200
- IS_UART_PARITY_BIT, 201
- IS_UART_STOP_BIT, 201
- IS_USERFLASH_STATUS, 221
- RCC_ADCClk_0, 167
- RCC_ADCClk_1, 167
- RCC_ADCClk_10, 167
- RCC_ADCClk_11, 167
- RCC_ADCClk_2, 167
- RCC_ADCClk_3, 167
- RCC_ADCClk_4, 167
- RCC_ADCClk_5, 167

- RCC_ADCClk_6, [167](#)
- RCC_ADCClk_7, [167](#)
- RCC_ADCClk_8, [167](#)
- RCC_ADCClk_9, [167](#)
- RCC_ADCClk_TypeDef, [167](#)
- RCC_PLLNO_Disable, [169](#)
- RCC_PLLNO_Div2, [169](#)
- RCC_PLLNO_Div4, [169](#)
- RCC_PLLNO_TypeDef, [169](#)
- RCC_PLLRef_ETH_25MHz, [169](#)
- RCC_PLLRef_TypeDef, [169](#)
- RCC_PLLRef_USB_60MHz, [169](#)
- RCC_PLLRef_USB_CLK, [169](#)
- RCC_PLLRef_XI_OSC, [169](#)
- RCC_PeriphClk_ADC, [168](#)
- RCC_PeriphClk_CMP, [167](#)
- RCC_PeriphClk_I2C0, [168](#)
- RCC_PeriphClk_I2C1, [168](#)
- RCC_PeriphClk_PWM0, [167](#)
- RCC_PeriphClk_PWM1, [167](#)
- RCC_PeriphClk_PWM2, [167](#)
- RCC_PeriphClk_PWM3, [167](#)
- RCC_PeriphClk_PWM4, [168](#)
- RCC_PeriphClk_PWM5, [168](#)
- RCC_PeriphClk_PWM6, [168](#)
- RCC_PeriphClk_PWM7, [168](#)
- RCC_PeriphClk_PWM8, [168](#)
- RCC_PeriphClk_QEP0, [167](#)
- RCC_PeriphClk_QEP1, [167](#)
- RCC_PeriphClk_TypeDef, [167](#)
- RCC_PeriphClk_WD, [168](#)
- RCC_PeriphRst_CAP0, [168](#)
- RCC_PeriphRst_CAP1, [168](#)
- RCC_PeriphRst_CAP2, [169](#)
- RCC_PeriphRst_CAP3, [169](#)
- RCC_PeriphRst_CAP4, [169](#)
- RCC_PeriphRst_CAP5, [169](#)
- RCC_PeriphRst_CMP, [169](#)
- RCC_PeriphRst_ETH, [168](#)
- RCC_PeriphRst_I2C0, [168](#)
- RCC_PeriphRst_I2C1, [168](#)
- RCC_PeriphRst_PWM0, [168](#)
- RCC_PeriphRst_PWM1, [168](#)
- RCC_PeriphRst_PWM2, [168](#)
- RCC_PeriphRst_PWM3, [168](#)
- RCC_PeriphRst_PWM4, [168](#)
- RCC_PeriphRst_PWM5, [168](#)
- RCC_PeriphRst_PWM6, [168](#)
- RCC_PeriphRst_PWM7, [168](#)
- RCC_PeriphRst_PWM8, [168](#)
- RCC_PeriphRst_QEP0, [168](#)
- RCC_PeriphRst_QEP1, [168](#)
- RCC_PeriphRst_SPI0, [168](#)
- RCC_PeriphRst_SPI1, [168](#)
- RCC_PeriphRst_SPI2, [168](#)
- RCC_PeriphRst_SPI3, [168](#)
- RCC_PeriphRst_Timer0, [168](#)
- RCC_PeriphRst_Timer1, [168](#)
- RCC_PeriphRst_Timer2, [168](#)
- RCC_PeriphRst_TypeDef, [168](#)
- RCC_PeriphRst_UART0, [168](#)
- RCC_PeriphRst_UART1, [168](#)
- RCC_PeriphRst_UART2, [168](#)
- RCC_PeriphRst_UART3, [168](#)
- RCC_PeriphRst_USB, [168](#)
- RCC_PeriphRst_WD, [168](#)
- RCC_SPIClk_SYSClk, [169](#)
- RCC_SPIClk_TypeDef, [169](#)
- RCC_SPIClk_USB_60MHz, [169](#)
- RCC_SPIClk_USB_CLK, [169](#)
- RCC_SPIClk_XI_OSC, [169](#)
- RCC_SysClk_CPE_Sel, [170](#)
- RCC_SysClk_ETH25MHz, [170](#)
- RCC_SysClk_PLL, [170](#)
- RCC_SysClk_PLLDIV, [170](#)
- RCC_SysClk_POR, [170](#)
- RCC_SysClk_TypeDef, [169](#)
- RCC_SysClk_USB60MHz, [170](#)
- RCC_SysClk_USB_CLK, [170](#)
- RCC_SysClk_XI_OSC, [170](#)
- RCC_UARTClk_SYSClk, [170](#)
- RCC_UARTClk_TypeDef, [170](#)
- RCC_UARTClk_USB_60MHz, [170](#)
- RCC_UARTClk_USB_CLK, [170](#)
- RCC_UARTClk_XI_OSC, [170](#)
- RCC_USBClk_TypeDef, [170](#)
- RCC_USBClk_USB_CLK, [170](#)
- RCC_USBClk_XI_OSC, [170](#)
- RCC_USBFreq_12MHz, [170](#)
- RCC_USBFreq_24MHz, [170](#)
- RCC_USBFreq_TypeDef, [170](#)
- RTC_Format_BCD, [186](#)
- RTC_Format_BIN, [186](#)
- RTC_Format_TypeDef, [186](#)
- RTC_Month_April, [187](#)
- RTC_Month_August, [187](#)
- RTC_Month_December, [187](#)
- RTC_Month_February, [187](#)
- RTC_Month_January, [187](#)
- RTC_Month_July, [187](#)
- RTC_Month_June, [187](#)
- RTC_Month_March, [187](#)
- RTC_Month_May, [187](#)
- RTC_Month_November, [187](#)
- RTC_Month_October, [187](#)
- RTC_Month_September, [187](#)
- RTC_Month_TypeDef, [186](#)
- RTC_Weekday_Friday, [187](#)
- RTC_Weekday_Monday, [187](#)
- RTC_Weekday_Saturday, [187](#)
- RTC_Weekday_Sunday, [187](#)
- RTC_Weekday_Thursday, [187](#)
- RTC_Weekday_Tuesday, [187](#)
- RTC_Weekday_TypeDef, [187](#)
- RTC_Weekday_Wednesday, [187](#)
- TIMER_ExtInput_CountClk, [189](#)

- TIMER_ExtInput_CountEn, [189](#)
- TIMER_ExtInput_Disable, [189](#)
- TIMER_ExtInput_TypeDef, [189](#)
- UART_DataWidth_5, [201](#)
- UART_DataWidth_6, [201](#)
- UART_DataWidth_7, [201](#)
- UART_DataWidth_8, [201](#)
- UART_DataWidth_TypeDef, [201](#)
- UART_Dir_Rx, [202](#)
- UART_Dir_Tx, [202](#)
- UART_Dir_Typedef, [201](#)
- UART_Error_Break, [202](#)
- UART_Error_Frame, [202](#)
- UART_Error_Overflow, [202](#)
- UART_Error_Parity, [202](#)
- UART_Error_Typedef, [202](#)
- UART_FIFOLevel_1_2, [202](#)
- UART_FIFOLevel_1_4, [202](#)
- UART_FIFOLevel_1_8, [202](#)
- UART_FIFOLevel_3_4, [202](#)
- UART_FIFOLevel_7_8, [202](#)
- UART_FIFOLevel_TypeDef, [202](#)
- UART_Flag_Busy, [202](#)
- UART_Flag_InvCTS, [202](#)
- UART_Flag_InvDCD, [202](#)
- UART_Flag_InvDSR, [202](#)
- UART_Flag_InvRI, [202](#)
- UART_Flag_RxFIFOEmpty, [202](#)
- UART_Flag_RxFIFOFull, [202](#)
- UART_Flag_TxFIFOEmpty, [202](#)
- UART_Flag_TxFIFOFull, [202](#)
- UART_Flag_Typedef, [202](#)
- UART_ITSource_ChangeCTS, [203](#)
- UART_ITSource_ChangeDCD, [203](#)
- UART_ITSource_ChangeDSR, [203](#)
- UART_ITSource_ChangeRI, [203](#)
- UART_ITSource_ErrorBreak, [203](#)
- UART_ITSource_ErrorFrame, [203](#)
- UART_ITSource_ErrorOverflow, [203](#)
- UART_ITSource_ErrorParity, [203](#)
- UART_ITSource_RecieveTimeout, [203](#)
- UART_ITSource_RxFIFOLevel, [203](#)
- UART_ITSource_TxFIFOLevel, [203](#)
- UART_ITSource_Typedef, [202](#)
- UART_ParityBit_Disable, [203](#)
- UART_ParityBit_Even, [203](#)
- UART_ParityBit_High, [203](#)
- UART_ParityBit_Low, [203](#)
- UART_ParityBit_Odd, [203](#)
- UART_ParityBit_TypeDef, [203](#)
- UART_StopBit_1, [203](#)
- UART_StopBit_2, [203](#)
- UART_StopBit_TypeDef, [203](#)
- USERFLASH_Status_Complete, [221](#)
- USERFLASH_Status_Error, [221](#)
- USERFLASH_Status_None, [221](#)
- USERFLASH_Status_TypeDef, [221](#)
- Управление сбросом, [184](#)
- RCC_PeriphRstCmd, [184](#)
- Управление тактированием, [175](#)
- RCC_PeriphClkCmd, [175](#)
- RCC_SysClkSel, [175](#)
- RCC_SysClkStatus, [176](#)
- Запись, [154](#)
- GPIO_Write, [154](#)
- GPIO_WriteBit, [154](#)
- GPIO_WriteMask, [154](#)
- ADC, [234](#)
- ADC_Average
 - ADC_Init_TypeDef, [373](#)
- ADC_Average_16
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_2
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_32
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_4
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_64
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_8
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_Disable
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Average_TypeDef
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_Channel_0
 - Маски каналов для измерений, [16](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_1
 - Маски каналов для измерений, [16](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_10
 - Маски каналов для измерений, [16](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_11
 - Маски каналов для измерений, [16](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_12
 - Маски каналов для измерений, [17](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_13
 - Маски каналов для измерений, [17](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_14
 - Маски каналов для измерений, [17](#)
 - niietcm4_adc.h, [425](#)
- ADC_Channel_15

- Маски каналов для измерений, [17](#)
- `niietcm4_adc.h`, [425](#)
- `ADC_Channel_16`
 - Маски каналов для измерений, [17](#)
 - `niietcm4_adc.h`, [425](#)
- `ADC_Channel_17`
 - Маски каналов для измерений, [17](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_18`
 - Маски каналов для измерений, [17](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_19`
 - Маски каналов для измерений, [17](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_2`
 - Маски каналов для измерений, [17](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_20`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_21`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_22`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_23`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_3`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [426](#)
- `ADC_Channel_4`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_5`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_6`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_7`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_8`
 - Маски каналов для измерений, [18](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_9`
 - Маски каналов для измерений, [19](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_All`
 - Маски каналов для измерений, [19](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channel_None`
 - Маски каналов для измерений, [19](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_Channels`
 - `ADC_SEQ_Init_TypeDef`, [374](#)
- `ADC_Cmd`
 - Функции, [37](#)
 - Приватные функции, [240](#)
 - `niietcm4_adc.c`, [403](#)
 - `niietcm4_adc.h`, [439](#)
- `ADC_DC_0`
 - Маски выбора цифровых компараторов, [20](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_DC_1`
 - Маски выбора цифровых компараторов, [20](#)
 - `niietcm4_adc.h`, [427](#)
- `ADC_DC_10`
 - Маски выбора цифровых компараторов, [20](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_11`
 - Маски выбора цифровых компараторов, [20](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_12`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_13`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_14`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_15`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_16`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_17`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_18`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [428](#)
- `ADC_DC_19`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [429](#)
- `ADC_DC_2`
 - Маски выбора цифровых компараторов, [21](#)
 - `niietcm4_adc.h`, [429](#)
- `ADC_DC_20`
 - Маски выбора цифровых компараторов, [22](#)
 - `niietcm4_adc.h`, [429](#)
- `ADC_DC_21`
 - Маски выбора цифровых компараторов, [22](#)
 - `niietcm4_adc.h`, [429](#)
- `ADC_DC_22`
 - Маски выбора цифровых компараторов, [22](#)
 - `niietcm4_adc.h`, [429](#)
- `ADC_DC_23`
 - Маски выбора цифровых компараторов, [22](#)
 - `niietcm4_adc.h`, [429](#)
- `ADC_DC_3`
 - Маски выбора цифровых компараторов, [22](#)

- niietcm4_adc.h, [429](#)
- ADC_DC_4
 - Маски выбора цифровых компараторов, [22](#)
 - niietcm4_adc.h, [429](#)
- ADC_DC_5
 - Маски выбора цифровых компараторов, [22](#)
 - niietcm4_adc.h, [429](#)
- ADC_DC_6
 - Маски выбора цифровых компараторов, [22](#)
 - niietcm4_adc.h, [429](#)
- ADC_DC_7
 - Маски выбора цифровых компараторов, [22](#)
 - niietcm4_adc.h, [430](#)
- ADC_DC_8
 - Маски выбора цифровых компараторов, [22](#)
 - niietcm4_adc.h, [430](#)
- ADC_DC_9
 - Маски выбора цифровых компараторов, [23](#)
 - niietcm4_adc.h, [430](#)
- ADC_DC_All
 - Маски выбора цифровых компараторов, [23](#)
 - niietcm4_adc.h, [430](#)
- ADC_DC_Channel
 - ADC_DC_Init_TypeDef, [372](#)
- ADC_DC_Channel_0
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_1
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_10
 - Типы, [32](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_11
 - Типы, [32](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_12
 - Типы, [32](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_13
 - Типы, [32](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_14
 - Типы, [32](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_15
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_16
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_17
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_18
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_19
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_2
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_20
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_21
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_22
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_23
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_3
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_4
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_5
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_6
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_7
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_8
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Channel_9
 - Типы, [32](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_None
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Channel_TypeDef
 - Типы, [32](#)
 - niietcm4_adc.h, [435](#)
- ADC_DC_Cmd
 - Функции, [38](#)
 - Приватные функции, [240](#)
 - niietcm4_adc.c, [404](#)
 - niietcm4_adc.h, [440](#)
- ADC_DC_Condition
 - ADC_DC_Init_TypeDef, [372](#)
- ADC_DC_Condition_High
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Condition_Low
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Condition_TypeDef

- Типы, [33](#)
- niietcm4_adc.h, [436](#)
- ADC_DC_Condition_Window
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_DeInit
 - Цифровые компараторы, [46](#)
 - Приватные функции, [240](#)
 - niietcm4_adc.c, [404](#)
 - niietcm4_adc.h, [440](#)
- ADC_DC_GetLastData
 - Функции, [38](#)
 - Приватные функции, [240](#)
 - niietcm4_adc.c, [404](#)
 - niietcm4_adc.h, [440](#)
- ADC_DC_ITCmd
 - Цифровые компараторы, [55](#)
 - Приватные функции, [241](#)
 - niietcm4_adc.c, [405](#)
 - niietcm4_adc.h, [442](#)
- ADC_DC_ITConfig
 - Цифровые компараторы, [55](#)
 - Приватные функции, [241](#)
 - niietcm4_adc.c, [405](#)
 - niietcm4_adc.h, [442](#)
- ADC_DC_ITGenCmd
 - Цифровые компараторы, [57](#)
 - Приватные функции, [242](#)
 - niietcm4_adc.c, [406](#)
 - niietcm4_adc.h, [442](#)
- ADC_DC_ITMaskCmd
 - Цифровые компараторы, [57](#)
 - Приватные функции, [242](#)
 - niietcm4_adc.c, [406](#)
 - niietcm4_adc.h, [444](#)
- ADC_DC_ITMaskedStatus
 - Цифровые компараторы, [57](#)
 - Приватные функции, [242](#)
 - niietcm4_adc.c, [406](#)
 - niietcm4_adc.h, [444](#)
- ADC_DC_ITRawStatus
 - Цифровые компараторы, [58](#)
 - Приватные функции, [243](#)
 - niietcm4_adc.c, [407](#)
 - niietcm4_adc.h, [444](#)
- ADC_DC_ITStatusClear
 - Цифровые компараторы, [58](#)
 - Приватные функции, [243](#)
 - niietcm4_adc.c, [407](#)
 - niietcm4_adc.h, [445](#)
- ADC_DC_Init
 - Цифровые компараторы, [46](#)
 - Приватные функции, [241](#)
 - niietcm4_adc.c, [405](#)
 - niietcm4_adc.h, [440](#)
- ADC_DC_Init_TypeDef, [371](#)
- ADC_DC_Channel, [372](#)
- ADC_DC_Condition, [372](#)
- ADC_DC_Mode, [372](#)
- ADC_DC_ThresholdHigh, [372](#)
- ADC_DC_ThresholdLow, [372](#)
- ADC_DC_Mode
 - ADC_DC_Init_TypeDef, [372](#)
- ADC_DC_Mode_Multiple
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Mode_MultipleHyst
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Mode_Single
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Mode_SingleHyst
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Mode_TypeDef
 - Типы, [33](#)
 - niietcm4_adc.h, [436](#)
- ADC_DC_Module_0
 - Типы, [33](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_1
 - Типы, [33](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_10
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_11
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_12
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_13
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_14
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_15
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_16
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_17
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_18
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_19
 - Типы, [34](#)
 - niietcm4_adc.h, [437](#)
- ADC_DC_Module_2
 - Типы, [33](#)

- niietcm4_adc.h, [437](#)
- ADC_DC_Module_20
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_21
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_22
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_23
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_3
 - Типы, [33](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_4
 - Типы, [33](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_5
 - Типы, [33](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_6
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_7
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_8
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_9
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_DC_Module_TypeDef
 - Типы, [33](#)
- niietcm4_adc.h, [436](#)
- ADC_DC_None
 - Маски выбора цифровых компараторов, [23](#)
- niietcm4_adc.h, [430](#)
- ADC_DC_StructInit
 - Цифровые компараторы, [46](#)
 - Приватные функции, [243](#)
- niietcm4_adc.c, [407](#)
 - niietcm4_adc.h, [445](#)
- ADC_DC_ThresholdHigh
 - ADC_DC_Init_TypeDef, [372](#)
- ADC_DC_ThresholdLow
 - ADC_DC_Init_TypeDef, [372](#)
- ADC_DC_TrigStatus
 - Функции, [38](#)
 - Приватные функции, [244](#)
- niietcm4_adc.c, [408](#)
 - niietcm4_adc.h, [445](#)
- ADC_DC_TrigStatusClear
 - Функции, [39](#)
 - Приватные функции, [244](#)
- niietcm4_adc.c, [408](#)
- niietcm4_adc.h, [445](#)
- ADC_DeInit
 - Модули АЦП, [44](#)
 - Приватные функции, [244](#)
- niietcm4_adc.c, [408](#)
 - niietcm4_adc.h, [446](#)
- ADC_Init
 - Модули АЦП, [44](#)
 - Приватные функции, [244](#)
- niietcm4_adc.c, [408](#)
 - niietcm4_adc.h, [446](#)
- ADC_Init_TypeDef, [373](#)
 - ADC_Average, [373](#)
 - ADC_Measure_A, [373](#)
 - ADC_Measure_B, [373](#)
 - ADC_Mode, [373](#)
 - ADC_Phase, [374](#)
 - ADC_Resolution, [374](#)
- ADC_Measure_A
 - ADC_Init_TypeDef, [373](#)
- ADC_Measure_B
 - ADC_Init_TypeDef, [373](#)
- ADC_Measure_Diff
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Measure_Single
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Measure_TypeDef
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Mode
 - ADC_Init_TypeDef, [373](#)
- ADC_Mode_Active
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Mode_Powerdown
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Mode_StandBy
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Mode_TypeDef
 - Типы, [34](#)
- niietcm4_adc.h, [437](#)
- ADC_Module_0
 - Типы, [35](#)
- niietcm4_adc.h, [438](#)
- ADC_Module_1
 - Типы, [35](#)
- niietcm4_adc.h, [438](#)
- ADC_Module_10
 - Типы, [35](#)
- niietcm4_adc.h, [438](#)
- ADC_Module_11
 - Типы, [35](#)
- niietcm4_adc.h, [438](#)
- ADC_Module_2

- Типы, [35](#)
- `niietcm4_adc.h`, [438](#)
- ADC_Module_3
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_4
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_5
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_6
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_7
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_8
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_9
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Module_TypeDef
 - Типы, [34](#)
 - `niietcm4_adc.h`, [437](#)
- ADC_Phase
 - ADC_Init_TypeDef, [374](#)
- ADC_Resolution
 - ADC_Init_TypeDef, [374](#)
- ADC_Resolution_10bit
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Resolution_12bit
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_Resolution_TypeDef
 - Типы, [35](#)
 - `niietcm4_adc.h`, [438](#)
- ADC_SEQ_0
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [430](#)
- ADC_SEQ_1
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [430](#)
- ADC_SEQ_2
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [430](#)
- ADC_SEQ_3
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [430](#)
- ADC_SEQ_4
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [431](#)
- ADC_SEQ_5
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [431](#)
- ADC_SEQ_6
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [431](#)
- ADC_SEQ_7
 - Маски выбора секвенсоров, [25](#)
 - `niietcm4_adc.h`, [431](#)
- ADC_SEQ_Cmd
 - Функции, [39](#)
 - Приватные функции, [246](#)
 - `niietcm4_adc.c`, [410](#)
 - `niietcm4_adc.h`, [446](#)
- ADC_SEQ_ConversionCount
 - ADC_SEQ_Init_TypeDef, [375](#)
- ADC_SEQ_ConversionDelay
 - ADC_SEQ_Init_TypeDef, [375](#)
- ADC_SEQ_DC
 - ADC_SEQ_Init_TypeDef, [375](#)
- ADC_SEQ_DMAMCmd
 - Конфигурация секвенсоров для DMA, [51](#)
 - Приватные функции, [246](#)
 - `niietcm4_adc.c`, [410](#)
 - `niietcm4_adc.h`, [447](#)
- ADC_SEQ_DMAConfig
 - Конфигурация секвенсоров для DMA, [51](#)
 - Приватные функции, [247](#)
 - `niietcm4_adc.c`, [411](#)
 - `niietcm4_adc.h`, [447](#)
- ADC_SEQ_DMAErrorStatus
 - Конфигурация секвенсоров для DMA, [51](#)
 - Приватные функции, [247](#)
 - `niietcm4_adc.c`, [411](#)
 - `niietcm4_adc.h`, [448](#)
- ADC_SEQ_DMAErrorStatusClear
 - Конфигурация секвенсоров для DMA, [53](#)
 - Приватные функции, [247](#)
 - `niietcm4_adc.c`, [411](#)
 - `niietcm4_adc.h`, [448](#)
- ADC_SEQ_DeInit
 - Приватные функции, [246](#)
 - Секвенсоры, [48](#)
 - `niietcm4_adc.c`, [410](#)
 - `niietcm4_adc.h`, [447](#)
- ADC_SEQ_FIFOEmptyStatus
 - Функции, [39](#)
 - Приватные функции, [248](#)
 - `niietcm4_adc.c`, [412](#)
 - `niietcm4_adc.h`, [448](#)
- ADC_SEQ_FIFOEmptyStatusClear
 - Функции, [40](#)
 - Приватные функции, [248](#)
 - `niietcm4_adc.c`, [412](#)
 - `niietcm4_adc.h`, [449](#)
- ADC_SEQ_FIFOFullStatus
 - Функции, [40](#)
 - Приватные функции, [248](#)
 - `niietcm4_adc.c`, [412](#)
 - `niietcm4_adc.h`, [449](#)
- ADC_SEQ_FIFOFullStatusClear
 - Функции, [40](#)

- Приватные функции, 249
- niietcm4_adc.c, 413
- niietcm4_adc.h, 449
- ADC_SEQ_FIFOLevel_1
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_FIFOLevel_16
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_FIFOLevel_2
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_FIFOLevel_32
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_FIFOLevel_4
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_FIFOLevel_8
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_FIFOLevel_TypeDef
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_GetConversionCount
 - Функции, 41
 - Приватные функции, 249
 - niietcm4_adc.c, 413
 - niietcm4_adc.h, 449
- ADC_SEQ_GetFIFOData
 - Функции, 41
 - Приватные функции, 249
 - niietcm4_adc.c, 413
 - niietcm4_adc.h, 451
- ADC_SEQ_GetFIFOLoad
 - Функции, 41
 - Приватные функции, 249
 - niietcm4_adc.c, 413
 - niietcm4_adc.h, 451
- ADC_SEQ_GetITCount
 - Приватные функции, 251
 - Секвенсоры, 59
 - niietcm4_adc.c, 415
 - niietcm4_adc.h, 451
- ADC_SEQ_ITCmd
 - Приватные функции, 251
 - Секвенсоры, 59
 - niietcm4_adc.c, 415
 - niietcm4_adc.h, 452
- ADC_SEQ_ITConfig
 - Приватные функции, 252
 - Секвенсоры, 60
 - niietcm4_adc.c, 416
 - niietcm4_adc.h, 452
- ADC_SEQ_ITCountRst
 - Приватные функции, 252
 - Секвенсоры, 60
 - niietcm4_adc.c, 416
- niietcm4_adc.h, 453
- ADC_SEQ_ITMaskedStatus
 - Приватные функции, 252
 - Секвенсоры, 60
 - niietcm4_adc.c, 416
 - niietcm4_adc.h, 453
- ADC_SEQ_ITRawStatus
 - Приватные функции, 253
 - Секвенсоры, 61
 - niietcm4_adc.c, 417
 - niietcm4_adc.h, 453
- ADC_SEQ_ITStatusClear
 - Приватные функции, 253
 - Секвенсоры, 61
 - niietcm4_adc.c, 417
 - niietcm4_adc.h, 454
- ADC_SEQ_Init
 - Приватные функции, 251
 - Секвенсоры, 48
 - niietcm4_adc.c, 415
 - niietcm4_adc.h, 452
- ADC_SEQ_Init_TypeDef, 374
 - ADC_Channels, 374
 - ADC_SEQ_ConversionCount, 375
 - ADC_SEQ_ConversionDelay, 375
 - ADC_SEQ_DC, 375
 - ADC_SEQ_SWReqEn, 375
 - ADC_SEQ_StartEvent, 375
- ADC_SEQ_Module_0
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_1
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_2
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_3
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_4
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_5
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_6
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_7
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_Module_TypeDef
 - Типы, 35
 - niietcm4_adc.h, 438
- ADC_SEQ_SWReq
 - Функции, 41
 - Приватные функции, 254

- niietcm4_adc.c, 418
 - niietcm4_adc.h, 454
- ADC_SEQ_SWReqEn
 - ADC_SEQ_Init_TypeDef, 375
- ADC_SEQ_StartEvent
 - ADC_SEQ_Init_TypeDef, 375
- ADC_SEQ_StartEvent_CMP0
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_CMP1
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_CMP2
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_Cycle
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_ITGPIO
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_PWM0
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_PWM1
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_PWM2
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_PWM3
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_PWM4
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_PWM5
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_SWReq
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_TIM
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StartEvent_TypeDef
 - Типы, 36
 - niietcm4_adc.h, 439
- ADC_SEQ_StructInit
 - Приватные функции, 253
 - Секвенсоры, 48
 - niietcm4_adc.c, 417
 - niietcm4_adc.h, 454
- ADC_StructInit
 - Модули АЦП, 44
 - Приватные функции, 254
 - niietcm4_adc.c, 418
 - niietcm4_adc.h, 454
- ALT_DATA
 - DMA_ConfigData_TypeDef, 384
- BOOTFLASH, 255
- BOOTFLASH_FullErase
 - Основная область флеш, 68
 - Приватные функции, 258
 - niietcm4_bootflash.c, 457
 - niietcm4_bootflash.h, 462
- BOOTFLASH_INFO_PAGE_SIZE_BYTES
 - Информационная область флеш, 64
 - niietcm4_bootflash.h, 461
- BOOTFLASH_INFO_PAGE_TOTAL
 - Информационная область флеш, 64
 - niietcm4_bootflash.h, 461
- BOOTFLASH_INFO_TOTAL_BYTES
 - Информационная область флеш, 64
 - niietcm4_bootflash.h, 461
- BOOTFLASH_ITCmd
 - Функции, 66
 - Приватные функции, 259
 - niietcm4_bootflash.c, 458
 - niietcm4_bootflash.h, 463
- BOOTFLASH_Info_PageErase
 - Информационная область флеш, 70
 - Приватные функции, 258
 - niietcm4_bootflash.c, 457
 - niietcm4_bootflash.h, 462
- BOOTFLASH_Info_Write
 - Информационная область флеш, 70
 - Приватные функции, 259
 - niietcm4_bootflash.c, 457
 - niietcm4_bootflash.h, 463
- BOOTFLASH_Init
 - Функции, 66
 - Приватные функции, 259
 - niietcm4_bootflash.c, 458
 - niietcm4_bootflash.h, 463
- BOOTFLASH_OperationStatus
 - Функции, 66
 - Приватные функции, 259
 - niietcm4_bootflash.c, 458
 - niietcm4_bootflash.h, 463
- BOOTFLASH_OperationStatusClear
 - Функции, 67
 - Приватные функции, 260
 - niietcm4_bootflash.c, 458
 - niietcm4_bootflash.h, 464
- BOOTFLASH_PAGE_SIZE_BYTES
 - Основная область флеш, 63
 - niietcm4_bootflash.h, 461
- BOOTFLASH_PAGE_TOTAL
 - Основная область флеш, 63
 - niietcm4_bootflash.h, 461
- BOOTFLASH_PageErase
 - Основная область флеш, 68
 - Приватные функции, 260
 - niietcm4_bootflash.c, 459
 - niietcm4_bootflash.h, 464

- BOOTFLASH_Status_Complete
 - Типы, [65](#)
 - nnietcm4_bootflash.h, [462](#)
- BOOTFLASH_Status_Error
 - Типы, [65](#)
 - nnietcm4_bootflash.h, [462](#)
- BOOTFLASH_Status_None
 - Типы, [65](#)
 - nnietcm4_bootflash.h, [462](#)
- BOOTFLASH_Status_TypeDef
 - Типы, [65](#)
 - nnietcm4_bootflash.h, [462](#)
- BOOTFLASH_TOTAL_BYTES
 - Основная область флеш, [63](#)
 - nnietcm4_bootflash.h, [462](#)
- BOOTFLASH_Write
 - Основная область флеш, [68](#)
 - Приватные функции, [260](#)
 - nnietcm4_bootflash.c, [459](#)
 - nnietcm4_bootflash.h, [464](#)
- BUFFERABLE
 - DMA_Protect_TypeDef, [386](#)
- Bit_CLEAR
 - Типы, [139](#)
 - nnietcm4_gpio.h, [560](#)
- Bit_SET
 - Типы, [139](#)
 - nnietcm4_gpio.h, [560](#)
- BitAction
 - Типы, [139](#)
 - nnietcm4_gpio.h, [560](#)
- CACHEABLE
 - DMA_Protect_TypeDef, [386](#)
- CAP, [261](#)
- CAP_Capture_Cmd
 - Приватные функции, [265](#)
 - Режим захвата, [89](#)
 - nnietcm4_cap.c, [467](#)
 - nnietcm4_cap.h, [487](#)
- CAP_Capture_GetCap0
 - Приватные функции, [265](#)
 - Режим захвата, [89](#)
 - nnietcm4_cap.c, [467](#)
 - nnietcm4_cap.h, [487](#)
- CAP_Capture_GetCap1
 - Приватные функции, [266](#)
 - Режим захвата, [90](#)
 - nnietcm4_cap.c, [467](#)
 - nnietcm4_cap.h, [488](#)
- CAP_Capture_GetCap2
 - Приватные функции, [266](#)
 - Режим захвата, [90](#)
 - nnietcm4_cap.c, [468](#)
 - nnietcm4_cap.h, [488](#)
- CAP_Capture_GetCap3
 - Приватные функции, [266](#)
 - Режим захвата, [90](#)
 - nnietcm4_cap.c, [468](#)
- nnietcm4_cap.h, [488](#)
- CAP_Capture_Init
 - Приватные функции, [266](#)
 - Режим захвата, [90](#)
 - nnietcm4_cap.c, [468](#)
 - nnietcm4_cap.h, [488](#)
- CAP_Capture_Init_TypeDef, [375](#)
- CAP_Capture_PolarityEvent0, [376](#)
- CAP_Capture_PolarityEvent1, [376](#)
- CAP_Capture_PolarityEvent2, [376](#)
- CAP_Capture_PolarityEvent3, [376](#)
- CAP_Capture_Prescale, [377](#)
- CAP_Capture_RstEvent0, [377](#)
- CAP_Capture_RstEvent1, [377](#)
- CAP_Capture_RstEvent2, [377](#)
- CAP_Capture_RstEvent3, [377](#)
- CAP_Capture_StopVal, [377](#)
- CAP_CaptureMode, [377](#)
- CAP_Capture_Mode_Cycle
 - Типы, [73](#)
 - nnietcm4_cap.h, [486](#)
- CAP_Capture_Mode_Single
 - Типы, [73](#)
 - nnietcm4_cap.h, [486](#)
- CAP_Capture_Mode_TypeDef
 - Типы, [73](#)
 - nnietcm4_cap.h, [486](#)
- CAP_Capture_Polarity_NegEdge
 - Типы, [73](#)
 - nnietcm4_cap.h, [486](#)
- CAP_Capture_Polarity_PosEdge
 - Типы, [73](#)
 - nnietcm4_cap.h, [486](#)
- CAP_Capture_Polarity_TypeDef
 - Типы, [73](#)
 - nnietcm4_cap.h, [486](#)
- CAP_Capture_PolarityEvent0
 - CAP_Capture_Init_TypeDef, [376](#)
- CAP_Capture_PolarityEvent1
 - CAP_Capture_Init_TypeDef, [376](#)
- CAP_Capture_PolarityEvent2
 - CAP_Capture_Init_TypeDef, [376](#)
- CAP_Capture_PolarityEvent3
 - CAP_Capture_Init_TypeDef, [376](#)
- CAP_Capture_Prescale
 - CAP_Capture_Init_TypeDef, [377](#)
- CAP_Capture_RstEvent0
 - CAP_Capture_Init_TypeDef, [377](#)
- CAP_Capture_RstEvent1
 - CAP_Capture_Init_TypeDef, [377](#)
- CAP_Capture_RstEvent2
 - CAP_Capture_Init_TypeDef, [377](#)
- CAP_Capture_RstEvent3
 - CAP_Capture_Init_TypeDef, [377](#)
- CAP_Capture_SetCap0
 - Приватные функции, [268](#)
 - Режим захвата, [92](#)
 - nnietcm4_cap.c, [469](#)

- niietcm4_cap.h, 490
- CAP_Capture_SetCap1
 - Приватные функции, 268
 - Режим захвата, 92
 - niietcm4_cap.c, 469
 - niietcm4_cap.h, 490
- CAP_Capture_SetCap2
 - Приватные функции, 268
 - Режим захвата, 92
 - niietcm4_cap.c, 469
 - niietcm4_cap.h, 490
- CAP_Capture_SetCap3
 - Приватные функции, 269
 - Режим захвата, 93
 - niietcm4_cap.c, 469
 - niietcm4_cap.h, 491
- CAP_Capture_StopVal
 - CAP_Capture_Init_TypeDef, 377
- CAP_Capture_StructInit
 - Приватные функции, 269
 - Режим захвата, 93
 - niietcm4_cap.c, 470
 - niietcm4_cap.h, 491
- CAP_CaptureMode
 - CAP_Capture_Init_TypeDef, 377
- CAP_DefInit
 - Конфигурация, 79
 - Приватные функции, 269
 - niietcm4_cap.c, 470
 - niietcm4_cap.h, 491
- CAP_GetShadowTimer
 - Конфигурация, 79
 - Приватные функции, 270
 - niietcm4_cap.c, 470
 - niietcm4_cap.h, 492
- CAP_GetTimer
 - Конфигурация, 80
 - Приватные функции, 270
 - niietcm4_cap.c, 471
 - niietcm4_cap.h, 492
- CAP_Halt
 - CAP_Init_TypeDef, 378
- CAP_Halt_Free
 - Типы, 74
 - niietcm4_cap.h, 486
- CAP_Halt_Stop
 - Типы, 73
 - niietcm4_cap.h, 486
- CAP_Halt_StopOnZero
 - Типы, 73
 - niietcm4_cap.h, 486
- CAP_Halt_TypeDef
 - Типы, 73
 - niietcm4_cap.h, 486
- CAP_ITCmd
 - Прерывания, 94
 - Приватные функции, 270
 - niietcm4_cap.c, 471
- niietcm4_cap.h, 492
- CAP_ITForceCmd
 - Прерывания, 94
 - Приватные функции, 271
 - niietcm4_cap.c, 472
 - niietcm4_cap.h, 493
- CAP_ITPendClear
 - Прерывания, 95
 - Приватные функции, 271
 - niietcm4_cap.c, 472
 - niietcm4_cap.h, 493
- CAP_ITPendStatus
 - Прерывания, 95
 - Приватные функции, 271
 - niietcm4_cap.c, 472
 - niietcm4_cap.h, 493
- CAP_ITSource_All
 - Маски источников прерываний, 76
 - niietcm4_cap.h, 483
- CAP_ITSource_CapEvent0
 - Маски источников прерываний, 76
 - niietcm4_cap.h, 483
- CAP_ITSource_CapEvent1
 - Маски источников прерываний, 76
 - niietcm4_cap.h, 483
- CAP_ITSource_CapEvent2
 - Маски источников прерываний, 76
 - niietcm4_cap.h, 483
- CAP_ITSource_CapEvent3
 - Маски источников прерываний, 76
 - niietcm4_cap.h, 483
- CAP_ITSource_GeneralInt
 - Маски источников прерываний, 76
 - niietcm4_cap.h, 484
- CAP_ITSource_TimerEqCompare
 - Маски источников прерываний, 77
 - niietcm4_cap.h, 484
- CAP_ITSource_TimerEqPeriod
 - Маски источников прерываний, 77
 - niietcm4_cap.h, 484
- CAP_ITSource_TimerOvf
 - Маски источников прерываний, 77
 - niietcm4_cap.h, 484
- CAP_ITStatus
 - Прерывания, 95
 - Приватные функции, 272
 - niietcm4_cap.c, 472
 - niietcm4_cap.h, 494
- CAP_ITStatusClear
 - Прерывания, 95
 - Приватные функции, 272
 - niietcm4_cap.c, 473
 - niietcm4_cap.h, 494
- CAP_Init
 - Конфигурация, 80
 - Приватные функции, 270
 - niietcm4_cap.c, 471
 - niietcm4_cap.h, 492

- CAP_Init_TypeDef, 378
 - CAP_Halt, 378
 - CAP_Mode, 378
 - CAP_SyncCmd, 378
 - CAP_SyncOut, 378
- CAP_Mode
 - CAP_Init_TypeDef, 378
- CAP_Mode_Capture
 - Типы, 74
 - niietcm4_cap.h, 487
- CAP_Mode_PWM
 - Типы, 74
 - niietcm4_cap.h, 487
- CAP_Mode_TypeDef
 - Типы, 74
 - niietcm4_cap.h, 486
- CAP_PWM_Compare
 - CAP_PWM_Init_TypeDef, 379
- CAP_PWM_GetCompare
 - Приватные функции, 272
 - Режим ШИМ, 84
 - niietcm4_cap.c, 473
 - niietcm4_cap.h, 494
- CAP_PWM_GetPeriod
 - Приватные функции, 272
 - Режим ШИМ, 84
 - niietcm4_cap.c, 473
 - niietcm4_cap.h, 494
- CAP_PWM_GetShadowCompare
 - Приватные функции, 274
 - Режим ШИМ, 85
 - niietcm4_cap.c, 473
 - niietcm4_cap.h, 496
- CAP_PWM_GetShadowPeriod
 - Приватные функции, 274
 - Режим ШИМ, 85
 - niietcm4_cap.c, 475
 - niietcm4_cap.h, 496
- CAP_PWM_Init
 - Приватные функции, 274
 - Режим ШИМ, 85
 - niietcm4_cap.c, 475
 - niietcm4_cap.h, 496
- CAP_PWM_Init_TypeDef, 379
 - CAP_PWM_Compare, 379
 - CAP_PWM_Period, 379
 - CAP_PWM_Polarity, 379
- CAP_PWM_Period
 - CAP_PWM_Init_TypeDef, 379
- CAP_PWM_Polarity
 - CAP_PWM_Init_TypeDef, 379
- CAP_PWM_Polarity_Neg
 - Типы, 74
 - niietcm4_cap.h, 487
- CAP_PWM_Polarity_Pos
 - Типы, 74
 - niietcm4_cap.h, 487
- CAP_PWM_Polarity_TypeDef
 - Типы, 74
 - niietcm4_cap.h, 487
- CAP_PWM_SetCompare
 - Приватные функции, 274
 - Режим ШИМ, 85
 - niietcm4_cap.c, 475
 - niietcm4_cap.h, 496
- CAP_PWM_SetPeriod
 - Приватные функции, 275
 - Режим ШИМ, 87
 - niietcm4_cap.c, 476
 - niietcm4_cap.h, 497
- CAP_PWM_SetShadowCompare
 - Приватные функции, 275
 - Режим ШИМ, 87
 - niietcm4_cap.c, 476
 - niietcm4_cap.h, 497
- CAP_PWM_SetShadowPeriod
 - Приватные функции, 275
 - Режим ШИМ, 87
 - niietcm4_cap.c, 476
 - niietcm4_cap.h, 497
- CAP_PWM_StructInit
 - Приватные функции, 276
 - Режим ШИМ, 88
 - niietcm4_cap.c, 476
 - niietcm4_cap.h, 498
- CAP_SetShadowTimer
 - Конфигурация, 80
 - Приватные функции, 276
 - niietcm4_cap.c, 478
 - niietcm4_cap.h, 498
- CAP_SetTimer
 - Конфигурация, 80
 - Приватные функции, 276
 - niietcm4_cap.c, 478
 - niietcm4_cap.h, 498
- CAP_StructInit
 - Конфигурация, 82
 - Приватные функции, 277
 - niietcm4_cap.c, 478
 - niietcm4_cap.h, 499
- CAP_SwSync
 - Конфигурация, 82
 - Приватные функции, 277
 - niietcm4_cap.c, 479
 - niietcm4_cap.h, 499
- CAP_SyncCmd
 - Конфигурация, 82
 - Приватные функции, 277
 - CAP_Init_TypeDef, 378
 - niietcm4_cap.c, 479
 - niietcm4_cap.h, 499
- CAP_SyncOut
 - CAP_Init_TypeDef, 378
- CAP_SyncOut_Bypass
 - Типы, 74
 - niietcm4_cap.h, 487

- CAP_SyncOut_Disable
 - Типы, [74](#)
 - niietcm4_cap.h, [487](#)
- CAP_SyncOut_TimerEqPeriod
 - Типы, [74](#)
 - niietcm4_cap.h, [487](#)
- CAP_SyncOut_TypeDef
 - Типы, [74](#)
 - niietcm4_cap.h, [487](#)
- CAP_TimerCmd
 - Конфигурация, [82](#)
 - Приватные функции, [277](#)
 - niietcm4_cap.c, [479](#)
 - niietcm4_cap.h, [499](#)
- CEMask
 - EXTMEM_Init_TypeDef, [387](#)
- CH
 - DMA_ConfigStruct_TypeDef, [384](#)
- CHANNEL_CFG
 - DMA_Channel_TypeDef, [380](#)
- CHANNEL_CFG_CYCLE_CTRL_Msk
 - Маски для CHANNEL_CFG, [98](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_CYCLE_CTRL_Pos
 - Маски для CHANNEL_CFG, [98](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_DST_INC_Msk
 - Маски для CHANNEL_CFG, [98](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_DST_INC_Pos
 - Маски для CHANNEL_CFG, [98](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_DST_PROT_CTRL_Msk
 - Маски для CHANNEL_CFG, [98](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_DST_PROT_CTRL_Pos
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_DST_SIZE_Msk
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_DST_SIZE_Pos
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_N_MINUS_1_Msk
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [514](#)
- CHANNEL_CFG_N_MINUS_1_Pos
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_NEXT_USEBURST_Msk
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_NEXT_USEBURST_Pos
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_R_POWER_Msk
 - Маски для CHANNEL_CFG, [99](#)
- niietcm4_dma.h, [515](#)
- CHANNEL_CFG_R_POWER_Pos
 - Маски для CHANNEL_CFG, [99](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_SRC_INC_Msk
 - Маски для CHANNEL_CFG, [100](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_SRC_INC_Pos
 - Маски для CHANNEL_CFG, [100](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_SRC_PROT_CTRL_Msk
 - Маски для CHANNEL_CFG, [100](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_SRC_PROT_CTRL_Pos
 - Маски для CHANNEL_CFG, [100](#)
 - niietcm4_dma.h, [515](#)
- CHANNEL_CFG_SRC_SIZE_Msk
 - Маски для CHANNEL_CFG, [100](#)
 - niietcm4_dma.h, [516](#)
- CHANNEL_CFG_SRC_SIZE_Pos
 - Маски для CHANNEL_CFG, [100](#)
 - niietcm4_dma.h, [516](#)
- CHANNEL_CFG_bit
 - DMA_Channel_TypeDef, [380](#)
- CYCLE_CTRL
 - _CHANNEL_CFG_bits, [369](#)
- DMA, [279](#)
- DMA_ArbitrationRate
 - DMA_ChannelInit_TypeDef, [381](#)
- DMA_ArbitrationRate_1
 - Типы, [113](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_1024
 - Типы, [114](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_128
 - Типы, [114](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_16
 - Типы, [113](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_2
 - Типы, [113](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_256
 - Типы, [114](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_32
 - Типы, [114](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_4
 - Типы, [113](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_512
 - Типы, [114](#)
 - niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_64
 - Типы, [114](#)

- niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_8
 - Типы, [113](#)
- niietcm4_dma.h, [523](#)
- DMA_ArbitrationRate_TypeDef
 - Типы, [113](#)
- niietcm4_dma.h, [523](#)
- DMA_BasePtrConfig
 - Конфигурация контроллера DMA, [121](#)
 - Приватные функции, [283](#)
- niietcm4_dma.c, [502](#)
 - niietcm4_dma.h, [525](#)
- DMA_Channel
 - DMA_Init_TypeDef, [385](#)
- DMA_Channel_0
 - Маски каналов по номеру, [107](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_1
 - Маски каналов по номеру, [107](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_10
 - Маски каналов по номеру, [107](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_11
 - Маски каналов по номеру, [107](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_12
 - Маски каналов по номеру, [107](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_13
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_14
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [516](#)
- DMA_Channel_15
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_16
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_17
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_18
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_19
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_2
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_20
 - Маски каналов по номеру, [108](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_21
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_22
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_23
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [517](#)
- DMA_Channel_3
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_4
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_5
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_6
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_7
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_8
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_9
 - Маски каналов по номеру, [109](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_ADCSEQ0
 - Маски каналов по имени, [103](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_ADCSEQ1
 - Маски каналов по имени, [103](#)
- niietcm4_dma.h, [518](#)
- DMA_Channel_ADCSEQ2
 - Маски каналов по имени, [103](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_ADCSEQ3
 - Маски каналов по имени, [103](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_ADCSEQ4
 - Маски каналов по имени, [103](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_ADCSEQ5
 - Маски каналов по имени, [104](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_ADCSEQ6
 - Маски каналов по имени, [104](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_ADCSEQ7
 - Маски каналов по имени, [104](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_All
 - Маски каналов DMA, [101](#)
- niietcm4_dma.h, [519](#)
- DMA_Channel_SPI0_RX
 - Маски каналов по имени, [104](#)
- niietcm4_dma.h, [519](#)

- DMA_Channel_SPI0_TX
 - Маски каналов по имени, [104](#)
 - niietcm4_dma.h, [519](#)
- DMA_Channel_SPI1_RX
 - Маски каналов по имени, [104](#)
 - niietcm4_dma.h, [519](#)
- DMA_Channel_SPI1_TX
 - Маски каналов по имени, [104](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_SPI2_RX
 - Маски каналов по имени, [104](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_SPI2_TX
 - Маски каналов по имени, [104](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_SPI3_RX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_SPI3_TX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_TypeDef, [380](#)
 - CHANNEL_CFG, [380](#)
 - CHANNEL_CFG_bit, [380](#)
 - DST_DATA_END, [380](#)
 - SRC_DATA_END, [380](#)
- DMA_Channel_UART0_RX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_UART0_TX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_UART1_RX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_UART1_TX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [520](#)
- DMA_Channel_UART2_RX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [521](#)
- DMA_Channel_UART2_TX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [521](#)
- DMA_Channel_UART3_RX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [521](#)
- DMA_Channel_UART3_TX
 - Маски каналов по имени, [105](#)
 - niietcm4_dma.h, [521](#)
- DMA_ChannelDeInit
 - Инициализация каналов DMA, [117](#)
 - Приватные функции, [284](#)
 - niietcm4_dma.c, [503](#)
 - niietcm4_dma.h, [525](#)
- DMA_ChannelEnable
 - DMA_Init_TypeDef, [385](#)
- DMA_ChannelEnableCmd
 - Конфигурация контроллера DMA, [121](#)
 - Приватные функции, [284](#)
 - niietcm4_dma.c, [503](#)
 - niietcm4_dma.h, [525](#)
- DMA_ChannelInit
 - Инициализация каналов DMA, [117](#)
 - Приватные функции, [284](#)
 - niietcm4_dma.c, [503](#)
 - niietcm4_dma.h, [526](#)
- DMA_ChannelInit_TypeDef, [381](#)
 - DMA_ArbitrationRate, [381](#)
 - DMA_DstDataEndPtr, [381](#)
 - DMA_DstDataInc, [381](#)
 - DMA_DstDataSize, [382](#)
 - DMA_DstProtect, [382](#)
 - DMA_Mode, [382](#)
 - DMA_NextUseburst, [382](#)
 - DMA_SrcDataEndPtr, [382](#)
 - DMA_SrcDataInc, [382](#)
 - DMA_SrcDataSize, [382](#)
 - DMA_SrcProtect, [383](#)
 - DMA_TransfersTotal, [383](#)
- DMA_ChannelStructInit
 - Инициализация каналов DMA, [118](#)
 - Приватные функции, [285](#)
 - niietcm4_dma.c, [504](#)
 - niietcm4_dma.h, [526](#)
- DMA_ClearErrorStatus
 - Приватные функции, [285](#)
 - Статусная информация, [125](#)
 - niietcm4_dma.c, [504](#)
 - niietcm4_dma.h, [527](#)
- DMA_ConfigData_TypeDef, [383](#)
 - ALT_DATA, [384](#)
 - PRM_DATA, [384](#)
 - RESERVED0, [384](#)
 - RESERVED1, [384](#)
- DMA_ConfigStruct_TypeDef, [384](#)
 - CH, [384](#)
- DMA_DataInc_16
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DataInc_32
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DataInc_8
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DataInc_Disable
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DataInc_TypeDef
 - Типы, [114](#)
 - niietcm4_dma.h, [523](#)
- DMA_DataSize_16
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DataSize_32

- Типы, [114](#)
- niietcm4_dma.h, [524](#)
- DMA_DataSize_8
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DataSize_TypeDef
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_DeInit
 - Инициализация контроллера DMA, [119](#)
 - Приватные функции, [285](#)
 - niietcm4_dma.c, [504](#)
 - niietcm4_dma.h, [527](#)
- DMA_DstDataEndPtr
 - DMA_ChannelInit_TypeDef, [381](#)
- DMA_DstDataInc
 - DMA_ChannelInit_TypeDef, [381](#)
- DMA_DstDataSize
 - DMA_ChannelInit_TypeDef, [382](#)
- DMA_DstProtect
 - DMA_ChannelInit_TypeDef, [382](#)
- DMA_ErrorStatus
 - Приватные функции, [287](#)
 - Статусная информация, [125](#)
 - niietcm4_dma.c, [506](#)
 - niietcm4_dma.h, [527](#)
- DMA_HighPriority
 - DMA_Init_TypeDef, [385](#)
- DMA_HighPriorityCmd
 - Конфигурация контроллера DMA, [122](#)
 - Приватные функции, [287](#)
 - niietcm4_dma.c, [506](#)
 - niietcm4_dma.h, [527](#)
- DMA_Init
 - Инициализация контроллера DMA, [119](#)
 - Приватные функции, [287](#)
 - niietcm4_dma.c, [506](#)
 - niietcm4_dma.h, [528](#)
- DMA_Init_TypeDef, [385](#)
 - DMA_Channel, [385](#)
 - DMA_ChannelEnable, [385](#)
 - DMA_HighPriority, [385](#)
 - DMA_PrmAlt, [385](#)
 - DMA_Protection, [385](#)
 - DMA_ReqMask, [386](#)
 - DMA_UseBurst, [386](#)
- DMA_MasterEnableCmd
 - Конфигурация контроллера DMA, [122](#)
 - Приватные функции, [288](#)
 - niietcm4_dma.c, [507](#)
 - niietcm4_dma.h, [528](#)
- DMA_MasterEnableStatus
 - Приватные функции, [288](#)
 - Статусная информация, [125](#)
 - niietcm4_dma.c, [507](#)
 - niietcm4_dma.h, [528](#)
- DMA_Mode
 - DMA_ChannelInit_TypeDef, [382](#)
- DMA_Mode_AltMemScatGath
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_AltPeriphScatGath
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_AutoReq
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_Basic
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_Disable
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_PingPong
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_PrmMemScatGath
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_PrmPeriphScatGath
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_Mode_TypeDef
 - Типы, [114](#)
 - niietcm4_dma.h, [524](#)
- DMA_NextUseburst
 - DMA_ChannelInit_TypeDef, [382](#)
- DMA_PrmAlt
 - DMA_Init_TypeDef, [385](#)
- DMA_PrmAltCmd
 - Конфигурация контроллера DMA, [122](#)
 - Приватные функции, [288](#)
 - niietcm4_dma.c, [507](#)
 - niietcm4_dma.h, [529](#)
- DMA_Protect_TypeDef, [386](#)
 - BUFFERABLE, [386](#)
 - CACHEABLE, [386](#)
 - PRIVELGED, [387](#)
- DMA_Protection
 - DMA_Init_TypeDef, [385](#)
- DMA_ProtectionConfig
 - Конфигурация контроллера DMA, [123](#)
 - Приватные функции, [288](#)
 - niietcm4_dma.c, [507](#)
 - niietcm4_dma.h, [529](#)
- DMA_ReqMask
 - DMA_Init_TypeDef, [386](#)
- DMA_ReqMaskCmd
 - Конфигурация контроллера DMA, [123](#)
 - Приватные функции, [289](#)
 - niietcm4_dma.c, [508](#)
 - niietcm4_dma.h, [529](#)
- DMA_SWRequestCmd
 - Конфигурация контроллера DMA, [123](#)
 - Приватные функции, [290](#)
 - niietcm4_dma.c, [509](#)

- niietcm4_dma.h, 530
- DMA_SrcDataEndPtr
 - DMA_ChannelInit_TypeDef, 382
- DMA_SrcDataInc
 - DMA_ChannelInit_TypeDef, 382
- DMA_SrcDataSize
 - DMA_ChannelInit_TypeDef, 382
- DMA_SrcProtect
 - DMA_ChannelInit_TypeDef, 383
- DMA_State_Done
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_Free
 - Типы, 115
 - niietcm4_dma.h, 524
- DMA_State_Pause
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_PeriphScatGath
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_ReadConfigData
 - Типы, 115
 - niietcm4_dma.h, 524
- DMA_State_ReadDstDataEndPtr
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_ReadSrcData
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_ReadSrcDataEndPtr
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_TypeDef
 - Типы, 114
 - niietcm4_dma.h, 524
- DMA_State_WaitReq
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_WriteConfigData
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_State_WriteDstData
 - Типы, 115
 - niietcm4_dma.h, 525
- DMA_StateStatus
 - Приватные функции, 289
 - Статусная информация, 125
 - niietcm4_dma.c, 508
 - niietcm4_dma.h, 530
- DMA_StructInit
 - Инициализация контроллера DMA, 119
 - Приватные функции, 289
 - niietcm4_dma.c, 508
 - niietcm4_dma.h, 530
- DMA_TransfersTotal
 - DMA_ChannelInit_TypeDef, 383
- DMA_UseBurst
 - DMA_Init_TypeDef, 386
- DMA_UseBurstCmd
 - Конфигурация контроллера DMA, 124
 - Приватные функции, 290
 - niietcm4_dma.c, 509
 - niietcm4_dma.h, 530
- DMA_WaitOnReqStatus
 - Приватные функции, 290
 - Статусная информация, 126
 - niietcm4_dma.c, 509
 - niietcm4_dma.h, 531
- DST_DATA_END
 - DMA_Channel_TypeDef, 380
- DST_INC
 - _CHANNEL_CFG_bits, 369
- DST_PROT_BUFFERABLE
 - _CHANNEL_CFG_bits, 370
- DST_PROT_CACHEABLE
 - _CHANNEL_CFG_bits, 370
- DST_PROT_PRIVILEGED
 - _CHANNEL_CFG_bits, 370
- DST_SIZE
 - _CHANNEL_CFG_bits, 370
- EXT_MEM_CFG_Reset_Value
 - Начальные значения регистров, 294
 - niietcm4_extmem.c, 532
- EXT_OSC_VALUE
 - Настройка драйвера, 363
 - niietcm4.h, 399
- EXTMEM, 291
- EXTMEM_CEMask_Addr_11
 - Маски адреса, 128
 - niietcm4_extmem.h, 535
- EXTMEM_CEMask_Addr_11_19
 - Маски адреса, 128
 - niietcm4_extmem.h, 535
- EXTMEM_CEMask_Addr_12
 - Маски адреса, 128
 - niietcm4_extmem.h, 535
- EXTMEM_CEMask_Addr_13
 - Маски адреса, 128
 - niietcm4_extmem.h, 535
- EXTMEM_CEMask_Addr_14
 - Маски адреса, 128
 - niietcm4_extmem.h, 535
- EXTMEM_CEMask_Addr_15
 - Маски адреса, 128
 - niietcm4_extmem.h, 535
- EXTMEM_CEMask_Addr_16
 - Маски адреса, 128
 - niietcm4_extmem.h, 536
- EXTMEM_CEMask_Addr_17
 - Маски адреса, 129
 - niietcm4_extmem.h, 536
- EXTMEM_CEMask_Addr_18
 - Маски адреса, 129
 - niietcm4_extmem.h, 536
- EXTMEM_CEMask_Addr_19

- Маски адреса, [129](#)
- niietcm4_extmem.h, [536](#)
- EXTMEM_DeInit
 - Функции, [134](#)
 - Приватные функции, [295](#)
 - niietcm4_extmem.c, [532](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_Init
 - Функции, [134](#)
 - Приватные функции, [295](#)
 - niietcm4_extmem.c, [532](#)
 - niietcm4_extmem.h, [539](#)
- EXTMEM_Init_TypeDef, [387](#)
 - CEMask, [387](#)
 - EXTMEM_RWWaitState, [388](#)
 - EXTMEM_ReadWaitState, [387](#)
 - EXTMEM_Width, [388](#)
 - EXTMEM_WriteWaitState, [388](#)
- EXTMEM_RWWaitState
 - EXTMEM_Init_TypeDef, [388](#)
- EXTMEM_RWWaitState_1
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_2
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_3
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_4
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_5
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_6
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_7
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_8
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_RWWaitState_TypeDef
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_ReadWaitState
 - EXTMEM_Init_TypeDef, [387](#)
- EXTMEM_ReadWaitState_1
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_2
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_3
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_4
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_5
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_6
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_7
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_ReadWaitState_8
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_ReadWaitState_TypeDef
 - Типы, [132](#)
 - niietcm4_extmem.h, [537](#)
- EXTMEM_StructInit
 - Функции, [134](#)
 - Приватные функции, [295](#)
 - niietcm4_extmem.c, [533](#)
 - niietcm4_extmem.h, [539](#)
- EXTMEM_Width
 - EXTMEM_Init_TypeDef, [388](#)
- EXTMEM_Width_16bit
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_Width_8bit
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_Width_TypeDef
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState
 - EXTMEM_Init_TypeDef, [388](#)
- EXTMEM_WriteWaitState_1
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_2
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_3
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_4
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_5
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_6
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_7
 - Типы, [133](#)
 - niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_8

- Типы, [133](#)
- niietcm4_extmem.h, [538](#)
- EXTMEM_WriteWaitState_TypeDef
 - Типы, [132](#)
 - niietcm4_extmem.h, [538](#)
- GPIO, [297](#)
- GPIO_AltFunc
 - GPIO_Init_TypeDef, [389](#)
- GPIO_AltFunc_1
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_AltFunc_2
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_AltFunc_3
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_AltFunc_TypeDef
 - Типы, [139](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_ClearBits
 - Битовые операции, [156](#)
 - Приватные функции, [305](#)
 - niietcm4_gpio.c, [544](#)
 - niietcm4_gpio.h, [562](#)
- GPIO_DATAOUT_Reset_Value
 - Начальные значения регистров, [300](#)
 - niietcm4_gpio.c, [541](#)
- GPIO_DeInit
 - Инициализация и деинициализация, [149](#)
 - Приватные функции, [306](#)
 - niietcm4_gpio.c, [544](#)
 - niietcm4_gpio.h, [562](#)
- GPIO_Dir
 - GPIO_Init_TypeDef, [389](#)
- GPIO_Dir_In
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_Dir_Out
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_Dir_TypeDef
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_GPIODEN0_Reset_Value
 - Начальные значения регистров, [300](#)
 - niietcm4_gpio.c, [541](#)
- GPIO_GPIODEN1_Reset_Value
 - Начальные значения регистров, [300](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIODEN2_Reset_Value
 - Начальные значения регистров, [300](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIODEN3_Reset_Value
 - Начальные значения регистров, [300](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIOODCTLx_Reset_Value
 - Начальные значения регистров, [301](#)
- niietcm4_gpio.c, [542](#)
- GPIO_GPIOODSCTLx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIOPCTLx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIOPUCTLx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIOQEx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [542](#)
- GPIO_GPIOQMx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [543](#)
- GPIO_GPIOQPx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [543](#)
- GPIO_GPIOSEx_Reset_Value
 - Начальные значения регистров, [301](#)
 - niietcm4_gpio.c, [543](#)
- GPIO_ITCmd
 - Прерывания, [160](#)
 - Приватные функции, [306](#)
 - niietcm4_gpio.c, [545](#)
 - niietcm4_gpio.h, [563](#)
- GPIO_ITConfig
 - Прерывания, [160](#)
 - Приватные функции, [308](#)
 - niietcm4_gpio.c, [545](#)
 - niietcm4_gpio.h, [563](#)
- GPIO_ITStatusClear
 - Прерывания, [160](#)
 - Приватные функции, [308](#)
 - niietcm4_gpio.c, [546](#)
 - niietcm4_gpio.h, [564](#)
- GPIO_Init
 - Инициализация и деинициализация, [149](#)
 - Приватные функции, [306](#)
 - niietcm4_gpio.c, [545](#)
 - niietcm4_gpio.h, [563](#)
- GPIO_Init_TypeDef, [388](#)
 - GPIO_AltFunc, [389](#)
 - GPIO_Dir, [389](#)
 - GPIO_Load, [389](#)
 - GPIO_Mode, [389](#)
 - GPIO_Out, [389](#)
 - GPIO_OutMode, [389](#)
 - GPIO_Pin, [389](#)
 - GPIO_PullUp, [390](#)
- GPIO_IntPol_Neg
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_IntPol_Pos
 - Типы, [140](#)
 - niietcm4_gpio.h, [560](#)
- GPIO_IntPol_TypeDef

- Типы, [140](#)
- `niietcm4_gpio.h`, [560](#)
- `GPIO_IntType_Edge`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [560](#)
- `GPIO_IntType_Level`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [560](#)
- `GPIO_IntType_TypeDef`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [560](#)
- `GPIO_Load`
 - `GPIO_Init_TypeDef`, [389](#)
- `GPIO_Load_16mA`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Load_8mA`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Load_TypeDef`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [560](#)
- `GPIO_Mode`
 - `GPIO_Init_TypeDef`, [389](#)
- `GPIO_Mode_AltFunc`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Mode_IO`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Mode_TypeDef`
 - Типы, [140](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Out`
 - `GPIO_Init_TypeDef`, [389](#)
- `GPIO_Out_Dis`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Out_En`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Out_TypeDef`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_OutMode`
 - `GPIO_Init_TypeDef`, [389](#)
- `GPIO_OutMode_OD`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_OutMode_PP`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_OutMode_TypeDef`
 - Типы, [141](#)
 - `niietcm4_gpio.h`, [561](#)
- `GPIO_Pin`
 - `GPIO_Init_TypeDef`, [389](#)
- `GPIO_Pin_0`
 - Маски пинов, [144](#)
 - `niietcm4_gpio.h`, [554](#)
- `GPIO_Pin_0_3`
 - Маски пинов, [144](#)
 - `niietcm4_gpio.h`, [554](#)
- `GPIO_Pin_0_7`
 - Маски пинов, [144](#)
 - `niietcm4_gpio.h`, [554](#)
- `GPIO_Pin_1`
 - Маски пинов, [144](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_10`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_11`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_12`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_12_15`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_13`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_14`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_15`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_2`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [555](#)
- `GPIO_Pin_3`
 - Маски пинов, [145](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_4`
 - Маски пинов, [146](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_4_7`
 - Маски пинов, [146](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_5`
 - Маски пинов, [146](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_6`
 - Маски пинов, [146](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_7`
 - Маски пинов, [146](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_8`
 - Маски пинов, [146](#)
 - `niietcm4_gpio.h`, [556](#)
- `GPIO_Pin_8_11`
 - Маски пинов, [146](#)

- niietcm4_gpio.h, 556
- GPIO_Pin_8_15
 - Маски пинов, 146
 - niietcm4_gpio.h, 556
- GPIO_Pin_9
 - Маски пинов, 146
 - niietcm4_gpio.h, 556
- GPIO_Pin_All
 - Маски пинов, 146
 - niietcm4_gpio.h, 557
- GPIO_PullUp
 - GPIO_Init_TypeDef, 390
- GPIO_PullUp_Dis
 - Типы, 141
 - niietcm4_gpio.h, 561
- GPIO_PullUp_En
 - Типы, 141
 - niietcm4_gpio.h, 561
- GPIO_PullUp_TypeDef
 - Типы, 141
 - niietcm4_gpio.h, 561
- GPIO_Qual_Dis
 - Типы, 141
 - niietcm4_gpio.h, 562
- GPIO_Qual_En
 - Типы, 141
 - niietcm4_gpio.h, 562
- GPIO_Qual_TypeDef
 - Типы, 141
 - niietcm4_gpio.h, 561
- GPIO_QualCmd
 - Фильтрация, 158
 - Приватные функции, 308
 - niietcm4_gpio.c, 546
 - niietcm4_gpio.h, 564
- GPIO_QualConfig
 - Фильтрация, 158
 - Приватные функции, 309
 - niietcm4_gpio.c, 546
 - niietcm4_gpio.h, 564
- GPIO_QualMode_3sample
 - Типы, 142
 - niietcm4_gpio.h, 562
- GPIO_QualMode_6sample
 - Типы, 142
 - niietcm4_gpio.h, 562
- GPIO_QualMode_TypeDef
 - Типы, 141
 - niietcm4_gpio.h, 562
- GPIO_Read
 - Чтение, 152
 - Приватные функции, 309
 - niietcm4_gpio.c, 548
 - niietcm4_gpio.h, 566
- GPIO_ReadBit
 - Чтение, 152
 - Приватные функции, 309
 - niietcm4_gpio.c, 548
- niietcm4_gpio.h, 566
- GPIO_ReadMask
 - Чтение, 152
 - Приватные функции, 310
 - niietcm4_gpio.c, 548
 - niietcm4_gpio.h, 566
- GPIO_Regs_A_C_E_G_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 543
- GPIO_Regs_B_D_F_H_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 543
- GPIO_Regs_GPIOA_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 543
- GPIO_Regs_GPIOB_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 543
- GPIO_Regs_GPIOC_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 543
- GPIO_Regs_GPIOD_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 544
- GPIO_Regs_GPIOE_Mask
 - Маски портов, 303
 - niietcm4_gpio.c, 544
- GPIO_Regs_GPIOF_Mask
 - Маски портов, 304
 - niietcm4_gpio.c, 544
- GPIO_Regs_GPIOG_Mask
 - Маски портов, 304
 - niietcm4_gpio.c, 544
- GPIO_Regs_GPIOH_Mask
 - Маски портов, 304
 - niietcm4_gpio.c, 544
- GPIO_SetBits
 - Битовые операции, 156
 - Приватные функции, 310
 - niietcm4_gpio.c, 549
 - niietcm4_gpio.h, 567
- GPIO_StructInit
 - Инициализация и деинициализация, 150
 - Приватные функции, 310
 - niietcm4_gpio.c, 549
 - niietcm4_gpio.h, 567
- GPIO_Sync_Dis
 - Типы, 142
 - niietcm4_gpio.h, 562
- GPIO_Sync_En
 - Типы, 142
 - niietcm4_gpio.h, 562
- GPIO_Sync_TypeDef
 - Типы, 142
 - niietcm4_gpio.h, 562
- GPIO_SyncCmd
 - Фильтрация, 159
 - Приватные функции, 311

- niietcm4_gpio.c, [549](#)
 - niietcm4_gpio.h, [567](#)
- GPIO_ToggleBits
 - Битовые операции, [156](#)
 - Приватные функции, [311](#)
 - niietcm4_gpio.c, [550](#)
 - niietcm4_gpio.h, [568](#)
- GPIO_Write
 - Приватные функции, [311](#)
 - Запись, [154](#)
 - niietcm4_gpio.c, [550](#)
 - niietcm4_gpio.h, [568](#)
- GPIO_WriteBit
 - Приватные функции, [312](#)
 - Запись, [154](#)
 - niietcm4_gpio.c, [550](#)
 - niietcm4_gpio.h, [568](#)
- GPIO_WriteMask
 - Приватные функции, [312](#)
 - Запись, [154](#)
 - niietcm4_gpio.c, [550](#)
 - niietcm4_gpio.h, [569](#)
- INT_OSC_VALUE
 - Настройка драйвера, [363](#)
 - niietcm4.h, [399](#)
- IS_ADC_AVERAGE
 - Типы, [28](#)
 - niietcm4_adc.h, [431](#)
- IS_ADC_DC_CHANNEL
 - Типы, [28](#)
 - niietcm4_adc.h, [431](#)
- IS_ADC_DC_CONDITION
 - Типы, [29](#)
 - niietcm4_adc.h, [432](#)
- IS_ADC_DC_MODE
 - Типы, [29](#)
 - niietcm4_adc.h, [432](#)
- IS_ADC_DC_MODULE
 - Типы, [29](#)
 - niietcm4_adc.h, [432](#)
- IS_ADC_MEASURE
 - Типы, [30](#)
 - niietcm4_adc.h, [433](#)
- IS_ADC_MODE
 - Типы, [30](#)
 - niietcm4_adc.h, [433](#)
- IS_ADC_MODULE
 - Типы, [30](#)
 - niietcm4_adc.h, [433](#)
- IS_ADC_RESOLUTION
 - Типы, [30](#)
 - niietcm4_adc.h, [433](#)
- IS_ADC_SEQ_FIFO_LEVEL
 - Типы, [31](#)
 - niietcm4_adc.h, [434](#)
- IS_ADC_SEQ_MODULE
 - Типы, [31](#)
 - niietcm4_adc.h, [434](#)
- IS_ADC_SEQ_START_EVENT
 - Типы, [31](#)
 - niietcm4_adc.h, [434](#)
- IS_BOOTFLASH_STATUS
 - Типы, [65](#)
 - niietcm4_bootflash.h, [462](#)
- IS_CAP_ALL_PERIPH
 - Типы, [365](#)
 - niietcm4.h, [399](#)
- IS_CAP_CAPTURE_MODE
 - Типы, [72](#)
 - niietcm4_cap.h, [484](#)
- IS_CAP_CAPTURE_POLARITY
 - Типы, [72](#)
 - niietcm4_cap.h, [484](#)
- IS_CAP_HALT
 - Типы, [72](#)
 - niietcm4_cap.h, [484](#)
- IS_CAP_IT_SOURCE_SINGLE
 - Маски источников прерываний, [77](#)
 - niietcm4_cap.h, [485](#)
- IS_CAP_MODE
 - Типы, [72](#)
 - niietcm4_cap.h, [485](#)
- IS_CAP_PWM_POLARITY
 - Типы, [72](#)
 - niietcm4_cap.h, [485](#)
- IS_CAP_SYNC_OUT
 - Типы, [73](#)
 - niietcm4_cap.h, [485](#)
- IS_DMA_ARBITRATION_RATE
 - Типы, [112](#)
 - niietcm4_dma.h, [521](#)
- IS_DMA_DATA_INC
 - Типы, [112](#)
 - niietcm4_dma.h, [521](#)
- IS_DMA_DATA_SIZE
 - Типы, [112](#)
 - niietcm4_dma.h, [522](#)
- IS_DMA_MODE
 - Типы, [113](#)
 - niietcm4_dma.h, [522](#)
- IS_DMA_STATE
 - Типы, [113](#)
 - niietcm4_dma.h, [522](#)
- IS_EXTMEM_READ_WAITSTATE
 - Типы, [130](#)
 - niietcm4_extmem.h, [536](#)
- IS_EXTMEM_RW_WAITSTATE
 - Типы, [131](#)
 - niietcm4_extmem.h, [536](#)
- IS_EXTMEM_WIDTH
 - Типы, [131](#)
 - niietcm4_extmem.h, [537](#)
- IS_EXTMEM_WRITE_WAITSTATE
 - Типы, [131](#)
 - niietcm4_extmem.h, [537](#)
- IS_GET_DMA_CHANNEL

- Маски каналов DMA, 101
- niietcm4_dma.h, 522
- IS_GET_GPIO_PIN
 - Маски пинов, 147
 - niietcm4_gpio.h, 557
- IS_GPIO_ALL_PERIPH
 - Типы, 365
 - niietcm4.h, 399
- IS_GPIO_ALT_FUNC
 - Типы, 137
 - niietcm4_gpio.h, 557
- IS_GPIO_DIR
 - Типы, 137
 - niietcm4_gpio.h, 557
- IS_GPIO_INT_POL
 - Типы, 137
 - niietcm4_gpio.h, 557
- IS_GPIO_INT_TYPE
 - Типы, 137
 - niietcm4_gpio.h, 558
- IS_GPIO_LOAD
 - Типы, 138
 - niietcm4_gpio.h, 558
- IS_GPIO_MODE
 - Типы, 138
 - niietcm4_gpio.h, 558
- IS_GPIO_OUT
 - Типы, 138
 - niietcm4_gpio.h, 558
- IS_GPIO_OUT_MODE
 - Типы, 138
 - niietcm4_gpio.h, 558
- IS_GPIO_PULLUP
 - Типы, 138
 - niietcm4_gpio.h, 559
- IS_GPIO_QUAL
 - Типы, 139
 - niietcm4_gpio.h, 559
- IS_GPIO_QUAL_MODE
 - Типы, 139
 - niietcm4_gpio.h, 559
- IS_GPIO_SYNC
 - Типы, 139
 - niietcm4_gpio.h, 559
- IS_RCC_ADC_CLK
 - Типы, 165
 - niietcm4_rcc.h, 583
- IS_RCC_PERIPH_CLK
 - Типы, 165
 - niietcm4_rcc.h, 583
- IS_RCC_PLL_NO
 - Типы, 165
 - niietcm4_rcc.h, 584
- IS_RCC_PLL_REF
 - Типы, 165
 - niietcm4_rcc.h, 584
- IS_RCC_SPI_CLK
 - Типы, 166
 - niietcm4_rcc.h, 584
- IS_RCC_SYS_CLK
 - Типы, 166
 - niietcm4_rcc.h, 584
- IS_RCC_UART_CLK
 - Типы, 166
 - niietcm4_rcc.h, 584
- IS_RCC_USB_CLK
 - Типы, 166
 - niietcm4_rcc.h, 585
- IS_RCC_USB_FREQ
 - Типы, 167
 - niietcm4_rcc.h, 585
- IS_RTC_FORMAT
 - Типы, 186
 - niietcm4_rtc.h, 599
- IS_RTC_MONTH
 - Типы, 186
 - niietcm4_rtc.h, 599
- IS_RTC_WEEKDAY
 - Типы, 186
 - niietcm4_rtc.h, 600
- IS_SPI_ALL_PERIPH
 - Типы, 366
 - niietcm4.h, 400
- IS_TIMER_ALL_PERIPH
 - Типы, 366
 - niietcm4.h, 400
- IS_TIMER_EXT_INPUT
 - Типы, 189
 - niietcm4_timer.h, 607
- IS_UART_ALL_PERIPH
 - Типы, 366
 - niietcm4.h, 400
- IS_UART_DATA_WIDTH
 - Типы, 199
 - niietcm4_uart.h, 624
- IS_UART_DIR
 - Типы, 199
 - niietcm4_uart.h, 624
- IS_UART_ERROR
 - Типы, 199
 - niietcm4_uart.h, 624
- IS_UART_FIFO_LEVEL
 - Типы, 200
 - niietcm4_uart.h, 625
- IS_UART_FLAG
 - Типы, 200
 - niietcm4_uart.h, 625
- IS_UART_GET_IT_SOURCE
 - Типы, 200
 - niietcm4_uart.h, 625
- IS_UART_PARITY_BIT
 - Типы, 201
 - niietcm4_uart.h, 626
- IS_UART_STOP_BIT
 - Типы, 201
 - niietcm4_uart.h, 626

- IS_USERFLASH_STATUS
 - Типы, [221](#)
 - niietcm4_userflash.h, [642](#)
- N_MINUS_1
 - _CHANNEL_CFG_bits, [370](#)
- NEXT_USEBURST
 - _CHANNEL_CFG_bits, [370](#)
- niietcm4.h, [397](#)
 - EXT_OSC_VALUE, [399](#)
 - INT_OSC_VALUE, [399](#)
 - IS_CAP_ALL_PERIPH, [399](#)
 - IS_GPIO_ALL_PERIPH, [399](#)
 - IS_SPI_ALL_PERIPH, [400](#)
 - IS_TIMER_ALL_PERIPH, [400](#)
 - IS_UART_ALL_PERIPH, [400](#)
- niietcm4_adc.c, [401](#)
 - ADC_Cmd, [403](#)
 - ADC_DC_Cmd, [404](#)
 - ADC_DC_DeInit, [404](#)
 - ADC_DC_GetLastData, [404](#)
 - ADC_DC_ITCmd, [405](#)
 - ADC_DC_ITConfig, [405](#)
 - ADC_DC_ITGenCmd, [406](#)
 - ADC_DC_ITMaskCmd, [406](#)
 - ADC_DC_ITMaskedStatus, [406](#)
 - ADC_DC_ITRawStatus, [407](#)
 - ADC_DC_ITStatusClear, [407](#)
 - ADC_DC_Init, [405](#)
 - ADC_DC_StructInit, [407](#)
 - ADC_DC_TrigStatus, [408](#)
 - ADC_DC_TrigStatusClear, [408](#)
 - ADC_DeInit, [408](#)
 - ADC_Init, [408](#)
 - ADC_SEQ_Cmd, [410](#)
 - ADC_SEQ_DMAMCmd, [410](#)
 - ADC_SEQ_DMAConfig, [411](#)
 - ADC_SEQ_DMAErrorStatus, [411](#)
 - ADC_SEQ_DMAErrorStatusClear, [411](#)
 - ADC_SEQ_DeInit, [410](#)
 - ADC_SEQ_FIFOEmptyStatus, [412](#)
 - ADC_SEQ_FIFOEmptyStatusClear, [412](#)
 - ADC_SEQ_FIFOFullStatus, [412](#)
 - ADC_SEQ_FIFOFullStatusClear, [413](#)
 - ADC_SEQ_GetConversionCount, [413](#)
 - ADC_SEQ_GetFIFOData, [413](#)
 - ADC_SEQ_GetFIFOLoad, [413](#)
 - ADC_SEQ_GetITCount, [415](#)
 - ADC_SEQ_ITCmd, [415](#)
 - ADC_SEQ_ITConfig, [416](#)
 - ADC_SEQ_ITCountRst, [416](#)
 - ADC_SEQ_ITMaskedStatus, [416](#)
 - ADC_SEQ_ITRawStatus, [417](#)
 - ADC_SEQ_ITStatusClear, [417](#)
 - ADC_SEQ_Init, [415](#)
 - ADC_SEQ_SWReq, [418](#)
 - ADC_SEQ_StructInit, [417](#)
 - ADC_StructInit, [418](#)
- niietcm4_adc.h, [418](#)
 - ADC_Average_16, [435](#)
 - ADC_Average_2, [435](#)
 - ADC_Average_32, [435](#)
 - ADC_Average_4, [435](#)
 - ADC_Average_64, [435](#)
 - ADC_Average_8, [435](#)
 - ADC_Average_Disable, [435](#)
 - ADC_Average_TypeDef, [435](#)
 - ADC_Channel_0, [425](#)
 - ADC_Channel_1, [425](#)
 - ADC_Channel_10, [425](#)
 - ADC_Channel_11, [425](#)
 - ADC_Channel_12, [425](#)
 - ADC_Channel_13, [425](#)
 - ADC_Channel_14, [425](#)
 - ADC_Channel_15, [425](#)
 - ADC_Channel_16, [425](#)
 - ADC_Channel_17, [426](#)
 - ADC_Channel_18, [426](#)
 - ADC_Channel_19, [426](#)
 - ADC_Channel_2, [426](#)
 - ADC_Channel_20, [426](#)
 - ADC_Channel_21, [426](#)
 - ADC_Channel_22, [426](#)
 - ADC_Channel_23, [426](#)
 - ADC_Channel_3, [426](#)
 - ADC_Channel_4, [427](#)
 - ADC_Channel_5, [427](#)
 - ADC_Channel_6, [427](#)
 - ADC_Channel_7, [427](#)
 - ADC_Channel_8, [427](#)
 - ADC_Channel_9, [427](#)
 - ADC_Channel_All, [427](#)
 - ADC_Channel_None, [427](#)
 - ADC_Cmd, [439](#)
 - ADC_DC_0, [427](#)
 - ADC_DC_1, [427](#)
 - ADC_DC_10, [428](#)
 - ADC_DC_11, [428](#)
 - ADC_DC_12, [428](#)
 - ADC_DC_13, [428](#)
 - ADC_DC_14, [428](#)
 - ADC_DC_15, [428](#)
 - ADC_DC_16, [428](#)
 - ADC_DC_17, [428](#)
 - ADC_DC_18, [428](#)
 - ADC_DC_19, [429](#)
 - ADC_DC_2, [429](#)
 - ADC_DC_20, [429](#)
 - ADC_DC_21, [429](#)
 - ADC_DC_22, [429](#)
 - ADC_DC_23, [429](#)
 - ADC_DC_3, [429](#)
 - ADC_DC_4, [429](#)
 - ADC_DC_5, [429](#)
 - ADC_DC_6, [429](#)
 - ADC_DC_7, [430](#)
 - ADC_DC_8, [430](#)

- ADC_DC_Channel_9, [430](#)
- ADC_DC_Channel_All, [430](#)
- ADC_DC_Channel_0, [435](#)
- ADC_DC_Channel_1, [435](#)
- ADC_DC_Channel_10, [436](#)
- ADC_DC_Channel_11, [436](#)
- ADC_DC_Channel_12, [436](#)
- ADC_DC_Channel_13, [436](#)
- ADC_DC_Channel_14, [436](#)
- ADC_DC_Channel_15, [436](#)
- ADC_DC_Channel_16, [436](#)
- ADC_DC_Channel_17, [436](#)
- ADC_DC_Channel_18, [436](#)
- ADC_DC_Channel_19, [436](#)
- ADC_DC_Channel_2, [435](#)
- ADC_DC_Channel_20, [436](#)
- ADC_DC_Channel_21, [436](#)
- ADC_DC_Channel_22, [436](#)
- ADC_DC_Channel_23, [436](#)
- ADC_DC_Channel_3, [435](#)
- ADC_DC_Channel_4, [435](#)
- ADC_DC_Channel_5, [435](#)
- ADC_DC_Channel_6, [435](#)
- ADC_DC_Channel_7, [435](#)
- ADC_DC_Channel_8, [435](#)
- ADC_DC_Channel_9, [436](#)
- ADC_DC_Channel_None, [436](#)
- ADC_DC_Channel_TypeDef, [435](#)
- ADC_DC_Cmd, [440](#)
- ADC_DC_Condition_High, [436](#)
- ADC_DC_Condition_Low, [436](#)
- ADC_DC_Condition_TypeDef, [436](#)
- ADC_DC_Condition_Window, [436](#)
- ADC_DC_DeInit, [440](#)
- ADC_DC_GetLastData, [440](#)
- ADC_DC_ITCmd, [442](#)
- ADC_DC_ITConfig, [442](#)
- ADC_DC_ITGenCmd, [442](#)
- ADC_DC_ITMaskCmd, [444](#)
- ADC_DC_ITMaskedStatus, [444](#)
- ADC_DC_ITRawStatus, [444](#)
- ADC_DC_ITStatusClear, [445](#)
- ADC_DC_Init, [440](#)
- ADC_DC_Mode_Multiple, [436](#)
- ADC_DC_Mode_MultipleHyst, [436](#)
- ADC_DC_Mode_Single, [436](#)
- ADC_DC_Mode_SingleHyst, [436](#)
- ADC_DC_Mode_TypeDef, [436](#)
- ADC_DC_Module_0, [437](#)
- ADC_DC_Module_1, [437](#)
- ADC_DC_Module_10, [437](#)
- ADC_DC_Module_11, [437](#)
- ADC_DC_Module_12, [437](#)
- ADC_DC_Module_13, [437](#)
- ADC_DC_Module_14, [437](#)
- ADC_DC_Module_15, [437](#)
- ADC_DC_Module_16, [437](#)
- ADC_DC_Module_17, [437](#)
- ADC_DC_Module_18, [437](#)
- ADC_DC_Module_19, [437](#)
- ADC_DC_Module_2, [437](#)
- ADC_DC_Module_20, [437](#)
- ADC_DC_Module_21, [437](#)
- ADC_DC_Module_22, [437](#)
- ADC_DC_Module_23, [437](#)
- ADC_DC_Module_3, [437](#)
- ADC_DC_Module_4, [437](#)
- ADC_DC_Module_5, [437](#)
- ADC_DC_Module_6, [437](#)
- ADC_DC_Module_7, [437](#)
- ADC_DC_Module_8, [437](#)
- ADC_DC_Module_9, [437](#)
- ADC_DC_Module_TypeDef, [436](#)
- ADC_DC_None, [430](#)
- ADC_DC_StructInit, [445](#)
- ADC_DC_TrigStatus, [445](#)
- ADC_DC_TrigStatusClear, [445](#)
- ADC_DeInit, [446](#)
- ADC_Init, [446](#)
- ADC_Measure_Diff, [437](#)
- ADC_Measure_Single, [437](#)
- ADC_Measure_TypeDef, [437](#)
- ADC_Mode_Active, [437](#)
- ADC_Mode_Powerdown, [437](#)
- ADC_Mode_StandBy, [437](#)
- ADC_Mode_TypeDef, [437](#)
- ADC_Module_0, [438](#)
- ADC_Module_1, [438](#)
- ADC_Module_10, [438](#)
- ADC_Module_11, [438](#)
- ADC_Module_2, [438](#)
- ADC_Module_3, [438](#)
- ADC_Module_4, [438](#)
- ADC_Module_5, [438](#)
- ADC_Module_6, [438](#)
- ADC_Module_7, [438](#)
- ADC_Module_8, [438](#)
- ADC_Module_9, [438](#)
- ADC_Module_TypeDef, [437](#)
- ADC_Resolution_10bit, [438](#)
- ADC_Resolution_12bit, [438](#)
- ADC_Resolution_TypeDef, [438](#)
- ADC_SEQ_0, [430](#)
- ADC_SEQ_1, [430](#)
- ADC_SEQ_2, [430](#)
- ADC_SEQ_3, [430](#)
- ADC_SEQ_4, [431](#)
- ADC_SEQ_5, [431](#)
- ADC_SEQ_6, [431](#)
- ADC_SEQ_7, [431](#)
- ADC_SEQ_Cmd, [446](#)
- ADC_SEQ_DMAMCmd, [447](#)
- ADC_SEQ_DMAConfig, [447](#)
- ADC_SEQ_DMAErrorStatus, [448](#)
- ADC_SEQ_DMAErrorStatusClear, [448](#)
- ADC_SEQ_DeInit, [447](#)

- ADC_SEQ_FIFOEmptyStatus, 448
- ADC_SEQ_FIFOEmptyStatusClear, 449
- ADC_SEQ_FIFOFullStatus, 449
- ADC_SEQ_FIFOFullStatusClear, 449
- ADC_SEQ_FIFOLevel_1, 438
- ADC_SEQ_FIFOLevel_16, 438
- ADC_SEQ_FIFOLevel_2, 438
- ADC_SEQ_FIFOLevel_32, 438
- ADC_SEQ_FIFOLevel_4, 438
- ADC_SEQ_FIFOLevel_8, 438
- ADC_SEQ_FIFOLevel_TypeDef, 438
- ADC_SEQ_GetConversionCount, 449
- ADC_SEQ_GetFIFOData, 451
- ADC_SEQ_GetFIFOLoad, 451
- ADC_SEQ_GetITCount, 451
- ADC_SEQ_ITCmd, 452
- ADC_SEQ_ITConfig, 452
- ADC_SEQ_ITCountRst, 453
- ADC_SEQ_ITMaskedStatus, 453
- ADC_SEQ_ITRawStatus, 453
- ADC_SEQ_ITStatusClear, 454
- ADC_SEQ_Init, 452
- ADC_SEQ_Module_0, 439
- ADC_SEQ_Module_1, 439
- ADC_SEQ_Module_2, 439
- ADC_SEQ_Module_3, 439
- ADC_SEQ_Module_4, 439
- ADC_SEQ_Module_5, 439
- ADC_SEQ_Module_6, 439
- ADC_SEQ_Module_7, 439
- ADC_SEQ_Module_TypeDef, 438
- ADC_SEQ_SWReq, 454
- ADC_SEQ_StartEvent_CMP0, 439
- ADC_SEQ_StartEvent_CMP1, 439
- ADC_SEQ_StartEvent_CMP2, 439
- ADC_SEQ_StartEvent_Cycle, 439
- ADC_SEQ_StartEvent_ITGPIO, 439
- ADC_SEQ_StartEvent_PWM0, 439
- ADC_SEQ_StartEvent_PWM1, 439
- ADC_SEQ_StartEvent_PWM2, 439
- ADC_SEQ_StartEvent_PWM3, 439
- ADC_SEQ_StartEvent_PWM4, 439
- ADC_SEQ_StartEvent_PWM5, 439
- ADC_SEQ_StartEvent_SWReq, 439
- ADC_SEQ_StartEvent_TIM, 439
- ADC_SEQ_StartEvent_TypeDef, 439
- ADC_SEQ_StructInit, 454
- ADC_StructInit, 454
- IS_ADC_AVERAGE, 431
- IS_ADC_DC_CHANNEL, 431
- IS_ADC_DC_CONDITION, 432
- IS_ADC_DC_MODE, 432
- IS_ADC_DC_MODULE, 432
- IS_ADC_MEASURE, 433
- IS_ADC_MODE, 433
- IS_ADC_MODULE, 433
- IS_ADC_RESOLUTION, 433
- IS_ADC_SEQ_FIFO_LEVEL, 434
- IS_ADC_SEQ_MODULE, 434
- IS_ADC_SEQ_START_EVENT, 434
- niietcm4_bootflash.c, 456
- BOOTFLASH_FullErase, 457
- BOOTFLASH_ITCmd, 458
- BOOTFLASH_Info_PageErase, 457
- BOOTFLASH_Info_Write, 457
- BOOTFLASH_Init, 458
- BOOTFLASH_OperationStatus, 458
- BOOTFLASH_OperationStatusClear, 458
- BOOTFLASH_PageErase, 459
- BOOTFLASH_Write, 459
- niietcm4_bootflash.h, 459
- BOOTFLASH_FullErase, 462
- BOOTFLASH_INFO_PAGE_SIZE_BYTES, 461
- BOOTFLASH_INFO_PAGE_TOTAL, 461
- BOOTFLASH_INFO_TOTAL_BYTES, 461
- BOOTFLASH_ITCmd, 463
- BOOTFLASH_Info_PageErase, 462
- BOOTFLASH_Info_Write, 463
- BOOTFLASH_Init, 463
- BOOTFLASH_OperationStatus, 463
- BOOTFLASH_OperationStatusClear, 464
- BOOTFLASH_PAGE_SIZE_BYTES, 461
- BOOTFLASH_PAGE_TOTAL, 461
- BOOTFLASH_PageErase, 464
- BOOTFLASH_Status_Complete, 462
- BOOTFLASH_Status_Error, 462
- BOOTFLASH_Status_None, 462
- BOOTFLASH_Status_TypeDef, 462
- BOOTFLASH_TOTAL_BYTES, 462
- BOOTFLASH_Write, 464
- IS_BOOTFLASH_STATUS, 462
- niietcm4_cap.c, 464
- CAP_Capture_Cmd, 467
- CAP_Capture_GetCap0, 467
- CAP_Capture_GetCap1, 467
- CAP_Capture_GetCap2, 468
- CAP_Capture_GetCap3, 468
- CAP_Capture_Init, 468
- CAP_Capture_SetCap0, 469
- CAP_Capture_SetCap1, 469
- CAP_Capture_SetCap2, 469
- CAP_Capture_SetCap3, 469
- CAP_Capture_StructInit, 470
- CAP_DeInit, 470
- CAP_GetShadowTimer, 470
- CAP_GetTimer, 471
- CAP_ITCmd, 471
- CAP_ITForceCmd, 472
- CAP_ITPendClear, 472
- CAP_ITPendStatus, 472
- CAP_ITStatus, 472
- CAP_ITStatusClear, 473
- CAP_Init, 471
- CAP_PWM_GetCompare, 473
- CAP_PWM_GetPeriod, 473

- CAP_PWM_GetShadowCompare, 473
- CAP_PWM_GetShadowPeriod, 475
- CAP_PWM_Init, 475
- CAP_PWM_SetCompare, 475
- CAP_PWM_SetPeriod, 476
- CAP_PWM_SetShadowCompare, 476
- CAP_PWM_SetShadowPeriod, 476
- CAP_PWM_StructInit, 476
- CAP_SetShadowTimer, 478
- CAP_SetTimer, 478
- CAP_StructInit, 478
- CAP_SwSync, 479
- CAP_SyncCmd, 479
- CAP_TimerCmd, 479
- niietcm4_cap.h, 479
 - CAP_Capture_Cmd, 487
 - CAP_Capture_GetCap0, 487
 - CAP_Capture_GetCap1, 488
 - CAP_Capture_GetCap2, 488
 - CAP_Capture_GetCap3, 488
 - CAP_Capture_Init, 488
 - CAP_Capture_Mode_Cycle, 486
 - CAP_Capture_Mode_Single, 486
 - CAP_Capture_Mode_TypeDef, 486
 - CAP_Capture_Polarity_NegEdge, 486
 - CAP_Capture_Polarity_PosEdge, 486
 - CAP_Capture_Polarity_TypeDef, 486
 - CAP_Capture_SetCap0, 490
 - CAP_Capture_SetCap1, 490
 - CAP_Capture_SetCap2, 490
 - CAP_Capture_SetCap3, 491
 - CAP_Capture_StructInit, 491
 - CAP_DeInit, 491
 - CAP_GetShadowTimer, 492
 - CAP_GetTimer, 492
 - CAP_Halt_Free, 486
 - CAP_Halt_Stop, 486
 - CAP_Halt_StopOnZero, 486
 - CAP_Halt_TypeDef, 486
 - CAP_ITCmd, 492
 - CAP_ITForceCmd, 493
 - CAP_ITPendClear, 493
 - CAP_ITPendStatus, 493
 - CAP_ITSource_All, 483
 - CAP_ITSource_CapEvent0, 483
 - CAP_ITSource_CapEvent1, 483
 - CAP_ITSource_CapEvent2, 483
 - CAP_ITSource_CapEvent3, 483
 - CAP_ITSource_GeneralInt, 484
 - CAP_ITSource_TimerEqCompare, 484
 - CAP_ITSource_TimerEqPeriod, 484
 - CAP_ITSource_TimerOvf, 484
 - CAP_ITStatus, 494
 - CAP_ITStatusClear, 494
 - CAP_Init, 492
 - CAP_Mode_Capture, 487
 - CAP_Mode_PWM, 487
 - CAP_Mode_TypeDef, 486
 - CAP_PWM_GetCompare, 494
 - CAP_PWM_GetPeriod, 494
 - CAP_PWM_GetShadowCompare, 496
 - CAP_PWM_GetShadowPeriod, 496
 - CAP_PWM_Init, 496
 - CAP_PWM_Polarity_Neg, 487
 - CAP_PWM_Polarity_Pos, 487
 - CAP_PWM_Polarity_TypeDef, 487
 - CAP_PWM_SetCompare, 496
 - CAP_PWM_SetPeriod, 497
 - CAP_PWM_SetShadowCompare, 497
 - CAP_PWM_SetShadowPeriod, 497
 - CAP_PWM_StructInit, 498
 - CAP_SetShadowTimer, 498
 - CAP_SetTimer, 498
 - CAP_StructInit, 499
 - CAP_SwSync, 499
 - CAP_SyncCmd, 499
 - CAP_SyncOut_Bypass, 487
 - CAP_SyncOut_Disable, 487
 - CAP_SyncOut_TimerEqPeriod, 487
 - CAP_SyncOut_TypeDef, 487
 - CAP_TimerCmd, 499
 - IS_CAP_CAPTURE_MODE, 484
 - IS_CAP_CAPTURE_POLARITY, 484
 - IS_CAP_HALT, 484
 - IS_CAP_IT_SOURCE_SINGLE, 485
 - IS_CAP_MODE, 485
 - IS_CAP_PWM_POLARITY, 485
 - IS_CAP_SYNC_OUT, 485
 - niietcm4_conf.h, 500
 - niietcm4_dma.c, 501
 - DMA_BasePtrConfig, 502
 - DMA_ChannelDeInit, 503
 - DMA_ChannelEnableCmd, 503
 - DMA_ChannelInit, 503
 - DMA_ChannelStructInit, 504
 - DMA_ClearErrorStatus, 504
 - DMA_DeInit, 504
 - DMA_ErrorStatus, 506
 - DMA_HighPriorityCmd, 506
 - DMA_Init, 506
 - DMA_MasterEnableCmd, 507
 - DMA_MasterEnableStatus, 507
 - DMA_PrmAltCmd, 507
 - DMA_ProtectionConfig, 507
 - DMA_ReqMaskCmd, 508
 - DMA_SWRequestCmd, 509
 - DMA_StateStatus, 508
 - DMA_StructInit, 508
 - DMA_UseBurstCmd, 509
 - DMA_WaitOnReqStatus, 509
 - niietcm4_dma.h, 510
 - CHANNEL_CFG_CYCLE_CTRL_Msk, 514
 - CHANNEL_CFG_CYCLE_CTRL_Pos, 514
 - CHANNEL_CFG_DST_INC_Msk, 514
 - CHANNEL_CFG_DST_INC_Pos, 514

- CHANNEL_CFG_DST_PROT_CTRL_ ←
Msk, [514](#)
- CHANNEL_CFG_DST_PROT_CTRL_ ←
Pos, [514](#)
- CHANNEL_CFG_DST_SIZE_Msk, [514](#)
- CHANNEL_CFG_DST_SIZE_Pos, [514](#)
- CHANNEL_CFG_N_MINUS_1_Msk, [514](#)
- CHANNEL_CFG_N_MINUS_1_Pos, [515](#)
- CHANNEL_CFG_NEXT_USEBURST_ ←
Msk, [515](#)
- CHANNEL_CFG_NEXT_USEBURST_ ←
Pos, [515](#)
- CHANNEL_CFG_R_POWER_Msk, [515](#)
- CHANNEL_CFG_R_POWER_Pos, [515](#)
- CHANNEL_CFG_SRC_INC_Msk, [515](#)
- CHANNEL_CFG_SRC_INC_Pos, [515](#)
- CHANNEL_CFG_SRC_PROT_CTRL_ ←
Msk, [515](#)
- CHANNEL_CFG_SRC_PROT_CTRL_ ←
Pos, [515](#)
- CHANNEL_CFG_SRC_SIZE_Msk, [516](#)
- CHANNEL_CFG_SRC_SIZE_Pos, [516](#)
- DMA_ArbitrationRate_1, [523](#)
- DMA_ArbitrationRate_1024, [523](#)
- DMA_ArbitrationRate_128, [523](#)
- DMA_ArbitrationRate_16, [523](#)
- DMA_ArbitrationRate_2, [523](#)
- DMA_ArbitrationRate_256, [523](#)
- DMA_ArbitrationRate_32, [523](#)
- DMA_ArbitrationRate_4, [523](#)
- DMA_ArbitrationRate_512, [523](#)
- DMA_ArbitrationRate_64, [523](#)
- DMA_ArbitrationRate_8, [523](#)
- DMA_ArbitrationRate_TypeDef, [523](#)
- DMA_BasePtrConfig, [525](#)
- DMA_Channel_0, [516](#)
- DMA_Channel_1, [516](#)
- DMA_Channel_10, [516](#)
- DMA_Channel_11, [516](#)
- DMA_Channel_12, [516](#)
- DMA_Channel_13, [516](#)
- DMA_Channel_14, [516](#)
- DMA_Channel_15, [517](#)
- DMA_Channel_16, [517](#)
- DMA_Channel_17, [517](#)
- DMA_Channel_18, [517](#)
- DMA_Channel_19, [517](#)
- DMA_Channel_2, [517](#)
- DMA_Channel_20, [517](#)
- DMA_Channel_21, [517](#)
- DMA_Channel_22, [517](#)
- DMA_Channel_23, [517](#)
- DMA_Channel_3, [518](#)
- DMA_Channel_4, [518](#)
- DMA_Channel_5, [518](#)
- DMA_Channel_6, [518](#)
- DMA_Channel_7, [518](#)
- DMA_Channel_8, [518](#)
- DMA_Channel_9, [518](#)
- DMA_Channel_ADCSEQ0, [518](#)
- DMA_Channel_ADCSEQ1, [518](#)
- DMA_Channel_ADCSEQ2, [519](#)
- DMA_Channel_ADCSEQ3, [519](#)
- DMA_Channel_ADCSEQ4, [519](#)
- DMA_Channel_ADCSEQ5, [519](#)
- DMA_Channel_ADCSEQ6, [519](#)
- DMA_Channel_ADCSEQ7, [519](#)
- DMA_Channel_All, [519](#)
- DMA_Channel_SPI0_RX, [519](#)
- DMA_Channel_SPI0_TX, [519](#)
- DMA_Channel_SPI1_RX, [519](#)
- DMA_Channel_SPI1_TX, [520](#)
- DMA_Channel_SPI2_RX, [520](#)
- DMA_Channel_SPI2_TX, [520](#)
- DMA_Channel_SPI3_RX, [520](#)
- DMA_Channel_SPI3_TX, [520](#)
- DMA_Channel_UART0_RX, [520](#)
- DMA_Channel_UART0_TX, [520](#)
- DMA_Channel_UART1_RX, [520](#)
- DMA_Channel_UART1_TX, [520](#)
- DMA_Channel_UART2_RX, [521](#)
- DMA_Channel_UART2_TX, [521](#)
- DMA_Channel_UART3_RX, [521](#)
- DMA_Channel_UART3_TX, [521](#)
- DMA_ChannelDeInit, [525](#)
- DMA_ChannelEnableCmd, [525](#)
- DMA_ChannelInit, [526](#)
- DMA_ChannelStructInit, [526](#)
- DMA_ClearErrorStatus, [527](#)
- DMA_DataInc_16, [524](#)
- DMA_DataInc_32, [524](#)
- DMA_DataInc_8, [524](#)
- DMA_DataInc_Disable, [524](#)
- DMA_DataInc_TypeDef, [523](#)
- DMA_DataSize_16, [524](#)
- DMA_DataSize_32, [524](#)
- DMA_DataSize_8, [524](#)
- DMA_DataSize_TypeDef, [524](#)
- DMA_DeInit, [527](#)
- DMA_ErrorStatus, [527](#)
- DMA_HighPriorityCmd, [527](#)
- DMA_Init, [528](#)
- DMA_MasterEnableCmd, [528](#)
- DMA_MasterEnableStatus, [528](#)
- DMA_Mode_AltMemScatGath, [524](#)
- DMA_Mode_AltPeriphScatGath, [524](#)
- DMA_Mode_AutoReq, [524](#)
- DMA_Mode_Basic, [524](#)
- DMA_Mode_Disable, [524](#)
- DMA_Mode_PingPong, [524](#)
- DMA_Mode_PrmMemScatGath, [524](#)
- DMA_Mode_PrmPeriphScatGath, [524](#)
- DMA_Mode_TypeDef, [524](#)
- DMA_PrmAltCmd, [529](#)
- DMA_ProtectionConfig, [529](#)
- DMA_ReqMaskCmd, [529](#)

- DMA_SWRequestCmd, 530
- DMA_State_Done, 525
- DMA_State_Free, 524
- DMA_State_Pause, 525
- DMA_State_PeriphScatGath, 525
- DMA_State_ReadConfigData, 524
- DMA_State_ReadDstDataEndPtr, 525
- DMA_State_ReadSrcData, 525
- DMA_State_ReadSrcDataEndPtr, 525
- DMA_State_TypeDef, 524
- DMA_State_WaitReq, 525
- DMA_State_WriteConfigData, 525
- DMA_State_WriteDstData, 525
- DMA_StateStatus, 530
- DMA_StructInit, 530
- DMA_UseBurstCmd, 530
- DMA_WaitOnReqStatus, 531
- IS_DMA_ARBITRATION_RATE, 521
- IS_DMA_DATA_INC, 521
- IS_DMA_DATA_SIZE, 522
- IS_DMA_MODE, 522
- IS_DMA_STATE, 522
- IS_GET_DMA_CHANNEL, 522
- niietcm4_extmem.c, 531
 - EXT_MEM_CFG_Reset_Value, 532
 - EXTMEM_DeInit, 532
 - EXTMEM_Init, 532
 - EXTMEM_StructInit, 533
- niietcm4_extmem.h, 533
 - EXTMEM_CEMask_Addr_11, 535
 - EXTMEM_CEMask_Addr_11_19, 535
 - EXTMEM_CEMask_Addr_12, 535
 - EXTMEM_CEMask_Addr_13, 535
 - EXTMEM_CEMask_Addr_14, 535
 - EXTMEM_CEMask_Addr_15, 535
 - EXTMEM_CEMask_Addr_16, 536
 - EXTMEM_CEMask_Addr_17, 536
 - EXTMEM_CEMask_Addr_18, 536
 - EXTMEM_CEMask_Addr_19, 536
 - EXTMEM_DeInit, 538
 - EXTMEM_Init, 539
 - EXTMEM_RWWaitState_1, 538
 - EXTMEM_RWWaitState_2, 538
 - EXTMEM_RWWaitState_3, 538
 - EXTMEM_RWWaitState_4, 538
 - EXTMEM_RWWaitState_5, 538
 - EXTMEM_RWWaitState_6, 538
 - EXTMEM_RWWaitState_7, 538
 - EXTMEM_RWWaitState_8, 538
 - EXTMEM_RWWaitState_TypeDef, 538
 - EXTMEM_ReadWaitState_1, 537
 - EXTMEM_ReadWaitState_2, 537
 - EXTMEM_ReadWaitState_3, 537
 - EXTMEM_ReadWaitState_4, 537
 - EXTMEM_ReadWaitState_5, 537
 - EXTMEM_ReadWaitState_6, 537
 - EXTMEM_ReadWaitState_7, 537
 - EXTMEM_ReadWaitState_8, 538
 - EXTMEM_ReadWaitState_TypeDef, 537
 - EXTMEM_StructInit, 539
 - EXTMEM_Width_16bit, 538
 - EXTMEM_Width_8bit, 538
 - EXTMEM_Width_TypeDef, 538
 - EXTMEM_WriteWaitState_1, 538
 - EXTMEM_WriteWaitState_2, 538
 - EXTMEM_WriteWaitState_3, 538
 - EXTMEM_WriteWaitState_4, 538
 - EXTMEM_WriteWaitState_5, 538
 - EXTMEM_WriteWaitState_6, 538
 - EXTMEM_WriteWaitState_7, 538
 - EXTMEM_WriteWaitState_8, 538
 - EXTMEM_WriteWaitState_TypeDef, 538
 - IS_EXTMEM_READ_WAITSTATE, 536
 - IS_EXTMEM_RW_WAITSTATE, 536
 - IS_EXTMEM_WIDTH, 537
 - IS_EXTMEM_WRITE_WAITSTATE, 537
- niietcm4_gpio.c, 539
 - GPIO_ClearBits, 544
 - GPIO_DATAOUT_Reset_Value, 541
 - GPIO_DeInit, 544
 - GPIO_GPIODEN0_Reset_Value, 541
 - GPIO_GPIODEN1_Reset_Value, 542
 - GPIO_GPIODEN2_Reset_Value, 542
 - GPIO_GPIODEN3_Reset_Value, 542
 - GPIO_GPIOODCTLx_Reset_Value, 542
 - GPIO_GPIOODSCTLx_Reset_Value, 542
 - GPIO_GPIOPCTLx_Reset_Value, 542
 - GPIO_GPIOPUCTLx_Reset_Value, 542
 - GPIO_GPIOQEx_Reset_Value, 542
 - GPIO_GPIOQMx_Reset_Value, 543
 - GPIO_GPIOQPx_Reset_Value, 543
 - GPIO_GPIOSEx_Reset_Value, 543
 - GPIO_ITCmd, 545
 - GPIO_ITConfig, 545
 - GPIO_ITStatusClear, 546
 - GPIO_Init, 545
 - GPIO_QualCmd, 546
 - GPIO_QualConfig, 546
 - GPIO_Read, 548
 - GPIO_ReadBit, 548
 - GPIO_ReadMask, 548
 - GPIO_Regs_A_C_E_G_Mask, 543
 - GPIO_Regs_B_D_F_H_Mask, 543
 - GPIO_Regs_GPIOA_Mask, 543
 - GPIO_Regs_GPIOB_Mask, 543
 - GPIO_Regs_GPIOC_Mask, 543
 - GPIO_Regs_GPIOD_Mask, 544
 - GPIO_Regs_GPIOE_Mask, 544
 - GPIO_Regs_GPIOF_Mask, 544
 - GPIO_Regs_GPIOG_Mask, 544
 - GPIO_Regs_GPIOH_Mask, 544
 - GPIO_SetBits, 549
 - GPIO_StructInit, 549
 - GPIO_SyncCmd, 549
 - GPIO_ToggleBits, 550
 - GPIO_Write, 550

- GPIO_WriteBit, 550
- GPIO_WriteMask, 550
- niietcm4_gpio.h, 551
 - Bit_CLEAR, 560
 - Bit_SET, 560
 - BitAction, 560
 - GPIO_AltFunc_1, 560
 - GPIO_AltFunc_2, 560
 - GPIO_AltFunc_3, 560
 - GPIO_AltFunc_TypeDef, 560
 - GPIO_ClearBits, 562
 - GPIO_DeInit, 562
 - GPIO_Dir_In, 560
 - GPIO_Dir_Out, 560
 - GPIO_Dir_TypeDef, 560
 - GPIO_ITCmd, 563
 - GPIO_ITConfig, 563
 - GPIO_ITStatusClear, 564
 - GPIO_Init, 563
 - GPIO_IntPol_Neg, 560
 - GPIO_IntPol_Pos, 560
 - GPIO_IntPol_TypeDef, 560
 - GPIO_IntType_Edge, 560
 - GPIO_IntType_Level, 560
 - GPIO_IntType_TypeDef, 560
 - GPIO_Load_16mA, 561
 - GPIO_Load_8mA, 561
 - GPIO_Load_TypeDef, 560
 - GPIO_Mode_AltFunc, 561
 - GPIO_Mode_IO, 561
 - GPIO_Mode_TypeDef, 561
 - GPIO_Out_Dis, 561
 - GPIO_Out_En, 561
 - GPIO_Out_TypeDef, 561
 - GPIO_OutMode_OD, 561
 - GPIO_OutMode_PP, 561
 - GPIO_OutMode_TypeDef, 561
 - GPIO_Pin_0, 554
 - GPIO_Pin_0_3, 554
 - GPIO_Pin_0_7, 554
 - GPIO_Pin_1, 555
 - GPIO_Pin_10, 555
 - GPIO_Pin_11, 555
 - GPIO_Pin_12, 555
 - GPIO_Pin_12_15, 555
 - GPIO_Pin_13, 555
 - GPIO_Pin_14, 555
 - GPIO_Pin_15, 555
 - GPIO_Pin_2, 555
 - GPIO_Pin_3, 556
 - GPIO_Pin_4, 556
 - GPIO_Pin_4_7, 556
 - GPIO_Pin_5, 556
 - GPIO_Pin_6, 556
 - GPIO_Pin_7, 556
 - GPIO_Pin_8, 556
 - GPIO_Pin_8_11, 556
 - GPIO_Pin_8_15, 556
 - GPIO_Pin_9, 556
 - GPIO_Pin_All, 557
 - GPIO_PullUp_Dis, 561
 - GPIO_PullUp_En, 561
 - GPIO_PullUp_TypeDef, 561
 - GPIO_Qual_Dis, 562
 - GPIO_Qual_En, 562
 - GPIO_Qual_TypeDef, 561
 - GPIO_QualCmd, 564
 - GPIO_QualConfig, 564
 - GPIO_QualMode_3sample, 562
 - GPIO_QualMode_6sample, 562
 - GPIO_QualMode_TypeDef, 562
 - GPIO_Read, 566
 - GPIO_ReadBit, 566
 - GPIO_ReadMask, 566
 - GPIO_SetBits, 567
 - GPIO_StructInit, 567
 - GPIO_Sync_Dis, 562
 - GPIO_Sync_En, 562
 - GPIO_Sync_TypeDef, 562
 - GPIO_SyncCmd, 567
 - GPIO_ToggleBits, 568
 - GPIO_Write, 568
 - GPIO_WriteBit, 568
 - GPIO_WriteMask, 569
 - IS_GET_GPIO_PIN, 557
 - IS_GPIO_ALT_FUNC, 557
 - IS_GPIO_DIR, 557
 - IS_GPIO_INT_POL, 557
 - IS_GPIO_INT_TYPE, 558
 - IS_GPIO_LOAD, 558
 - IS_GPIO_MODE, 558
 - IS_GPIO_OUT, 558
 - IS_GPIO_OUT_MODE, 558
 - IS_GPIO_PULLUP, 559
 - IS_GPIO_QUAL, 559
 - IS_GPIO_QUAL_MODE, 559
 - IS_GPIO_SYNC, 559
 - niietcm4_rcc.c, 569
 - RCC_ADCClkCmd, 571
 - RCC_ADCClkDivConfig, 572
 - RCC_PLL_CTRL_Reset_Value, 571
 - RCC_PLL_NF_Reset_Value, 571
 - RCC_PLL_NR_Reset_Value, 571
 - RCC_PLL_OD_Reset_Value, 571
 - RCC_PLLAutoConfig, 573
 - RCC_PLLDeInit, 574
 - RCC_PLLInit, 574
 - RCC_PLLPowerDownCmd, 575
 - RCC_PLLStructInit, 575
 - RCC_PeriphClkCmd, 572
 - RCC_PeriphRstCmd, 573
 - RCC_SPIClkCmd, 575
 - RCC_SPIClkDivConfig, 576
 - RCC_SPIClkSel, 576
 - RCC_SysClkDiv2Out, 576
 - RCC_SysClkSel, 577

- RCC_SysClkStatus, 577
- RCC_UARTClkCmd, 577
- RCC_UARTClkDivConfig, 578
- RCC_UARTClkSel, 578
- RCC_USBClkCmd, 578
- RCC_USBClkConfig, 579
- RCC_WaitClkChange, 579
- niietcm4_rcc.h, 579
- IS_RCC_ADC_CLK, 583
- IS_RCC_PERIPH_CLK, 583
- IS_RCC_PLL_NO, 584
- IS_RCC_PLL_REF, 584
- IS_RCC_SPI_CLK, 584
- IS_RCC_SYS_CLK, 584
- IS_RCC_UART_CLK, 584
- IS_RCC_USB_CLK, 585
- IS_RCC_USB_FREQ, 585
- RCC_ADCClk_0, 586
- RCC_ADCClk_1, 586
- RCC_ADCClk_10, 586
- RCC_ADCClk_11, 586
- RCC_ADCClk_2, 586
- RCC_ADCClk_3, 586
- RCC_ADCClk_4, 586
- RCC_ADCClk_5, 586
- RCC_ADCClk_6, 586
- RCC_ADCClk_7, 586
- RCC_ADCClk_8, 586
- RCC_ADCClk_9, 586
- RCC_ADCClk_TypeDef, 585
- RCC_ADCClkCmd, 589
- RCC_ADCClkDivConfig, 589
- RCC_CLK_CHANGE_TIMEOUT, 585
- RCC_CLK_PLL_STABLE_TIMEOUT, 585
- RCC_PLLAutoConfig, 591
- RCC_PLLDeInit, 591
- RCC_PLLInit, 591
- RCC_PLLNO_Disable, 587
- RCC_PLLNO_Div2, 587
- RCC_PLLNO_Div4, 587
- RCC_PLLNO_TypeDef, 587
- RCC_PLLPowerDownCmd, 592
- RCC_PLLRef_ETH_25MHz, 588
- RCC_PLLRef_TypeDef, 587
- RCC_PLLRef_USB_60MHz, 588
- RCC_PLLRef_USB_CLK, 588
- RCC_PLLRef_XI_OSC, 588
- RCC_PLLStructInit, 593
- RCC_PeriphClk_ADC, 586
- RCC_PeriphClk_CMP, 586
- RCC_PeriphClk_I2C0, 586
- RCC_PeriphClk_I2C1, 586
- RCC_PeriphClk_PWM0, 586
- RCC_PeriphClk_PWM1, 586
- RCC_PeriphClk_PWM2, 586
- RCC_PeriphClk_PWM3, 586
- RCC_PeriphClk_PWM4, 586
- RCC_PeriphClk_PWM5, 586
- RCC_PeriphClk_PWM6, 586
- RCC_PeriphClk_PWM7, 586
- RCC_PeriphClk_PWM8, 586
- RCC_PeriphClk_QEP0, 586
- RCC_PeriphClk_QEP1, 586
- RCC_PeriphClk_TypeDef, 586
- RCC_PeriphClk_WD, 586
- RCC_PeriphClkCmd, 590
- RCC_PeriphRst_CAP0, 587
- RCC_PeriphRst_CAP1, 587
- RCC_PeriphRst_CAP2, 587
- RCC_PeriphRst_CAP3, 587
- RCC_PeriphRst_CAP4, 587
- RCC_PeriphRst_CAP5, 587
- RCC_PeriphRst_CMP, 587
- RCC_PeriphRst_ETH, 587
- RCC_PeriphRst_I2C0, 586
- RCC_PeriphRst_I2C1, 587
- RCC_PeriphRst_PWM0, 587
- RCC_PeriphRst_PWM1, 587
- RCC_PeriphRst_PWM2, 587
- RCC_PeriphRst_PWM3, 587
- RCC_PeriphRst_PWM4, 587
- RCC_PeriphRst_PWM5, 587
- RCC_PeriphRst_PWM6, 587
- RCC_PeriphRst_PWM7, 587
- RCC_PeriphRst_PWM8, 587
- RCC_PeriphRst_QEP0, 587
- RCC_PeriphRst_QEP1, 587
- RCC_PeriphRst_SPI0, 587
- RCC_PeriphRst_SPI1, 587
- RCC_PeriphRst_SPI2, 587
- RCC_PeriphRst_SPI3, 587
- RCC_PeriphRst_Timer0, 587
- RCC_PeriphRst_Timer1, 587
- RCC_PeriphRst_Timer2, 587
- RCC_PeriphRst_TypeDef, 586
- RCC_PeriphRst_UART0, 587
- RCC_PeriphRst_UART1, 587
- RCC_PeriphRst_UART2, 587
- RCC_PeriphRst_UART3, 587
- RCC_PeriphRst_USB, 587
- RCC_PeriphRst_WD, 586
- RCC_PeriphRstCmd, 590
- RCC_SPIClk_SYSCLK, 588
- RCC_SPIClk_TypeDef, 588
- RCC_SPIClk_USB_60MHz, 588
- RCC_SPIClk_USB_CLK, 588
- RCC_SPIClk_XI_OSC, 588
- RCC_SPIClkCmd, 593
- RCC_SPIClkDivConfig, 593
- RCC_SPIClkSel, 594
- RCC_SysClk_CPE_Sel, 588
- RCC_SysClk_ETH25MHz, 588
- RCC_SysClk_PLL, 588
- RCC_SysClk_PLLDIV, 588
- RCC_SysClk_POR, 588

- RCC_SysClk_TypeDef, 588
- RCC_SysClk_USB60MHz, 588
- RCC_SysClk_USB_CLK, 588
- RCC_SysClk_XI_OSC, 588
- RCC_SysClkDiv2Out, 594
- RCC_SysClkSel, 594
- RCC_SysClkStatus, 595
- RCC_UARTClk_SYSCLK, 588
- RCC_UARTClk_TypeDef, 588
- RCC_UARTClk_USB_60MHz, 588
- RCC_UARTClk_USB_CLK, 588
- RCC_UARTClk_XI_OSC, 588
- RCC_UARTClkCmd, 595
- RCC_UARTClkDivConfig, 595
- RCC_UARTClkSel, 595
- RCC_USBClk_TypeDef, 588
- RCC_USBClk_USB_CLK, 589
- RCC_USBClk_XI_OSC, 589
- RCC_USBClkCmd, 596
- RCC_USBClkConfig, 596
- RCC_USBFreq_12MHz, 589
- RCC_USBFreq_24MHz, 589
- RCC_USBFreq_TypeDef, 589
- niietcm4_rtc.c, 596
- niietcm4_rtc.h, 597
 - IS_RTC_FORMAT, 599
 - IS_RTC_MONTH, 599
 - IS_RTC_WEEKDAY, 600
 - RTC_Format_BCD, 600
 - RTC_Format_BIN, 600
 - RTC_Format_TypeDef, 600
 - RTC_Month_April, 600
 - RTC_Month_August, 600
 - RTC_Month_December, 600
 - RTC_Month_February, 600
 - RTC_Month_January, 600
 - RTC_Month_July, 600
 - RTC_Month_June, 600
 - RTC_Month_March, 600
 - RTC_Month_May, 600
 - RTC_Month_November, 600
 - RTC_Month_October, 600
 - RTC_Month_September, 600
 - RTC_Month_TypeDef, 600
 - RTC_Weekday_Friday, 601
 - RTC_Weekday_Monday, 601
 - RTC_Weekday_Saturday, 601
 - RTC_Weekday_Sunday, 601
 - RTC_Weekday_Thursday, 601
 - RTC_Weekday_Tuesday, 601
 - RTC_Weekday_TypeDef, 600
 - RTC_Weekday_Wednesday, 601
- niietcm4_timer.c, 601
 - TIMER_Cmd, 602
 - TIMER_ExtInputConfig, 602
 - TIMER_FreqConfig, 603
 - TIMER_GetCounter, 603
 - TIMER_GetReload, 603
 - TIMER_ITCmd, 604
 - TIMER_ITStatus, 604
 - TIMER_ITStatusClear, 604
 - TIMER_PeriodConfig, 604
 - TIMER_SetCounter, 605
 - TIMER_SetReload, 605
- niietcm4_timer.h, 605
 - IS_TIMER_EXT_INPUT, 607
 - TIMER_Cmd, 608
 - TIMER_ExtInput_CountClk, 607
 - TIMER_ExtInput_CountEn, 607
 - TIMER_ExtInput_Disable, 607
 - TIMER_ExtInput_TypeDef, 607
 - TIMER_ExtInputConfig, 609
 - TIMER_FreqConfig, 609
 - TIMER_GetCounter, 609
 - TIMER_GetReload, 610
 - TIMER_ITCmd, 610
 - TIMER_ITStatus, 610
 - TIMER_ITStatusClear, 610
 - TIMER_PeriodConfig, 611
 - TIMER_SetCounter, 611
 - TIMER_SetReload, 611
- niietcm4_uart.c, 612
 - UART_BaudRateDivConfig, 614
 - UART_Break, 614
 - UART_Cmd, 614
 - UART_DMABlkOnErrCmd, 616
 - UART_DMACmd, 616
 - UART_DeInit, 614
 - UART_ErrorStatus, 616
 - UART_ErrorStatusClear, 617
 - UART_FlagStatus, 617
 - UART_ITCmd, 618
 - UART_ITFIFOLevelConfig, 618
 - UART_ITMaskedStatus, 618
 - UART_ITRawStatus, 619
 - UART_ITStatusClear, 619
 - UART_Init, 617
 - UART_ModemConfig, 619
 - UART_ModemStructInit, 620
 - UART_RecieveData, 620
 - UART_SendData, 620
 - UART_StructInit, 620
- niietcm4_uart.h, 621
 - IS_UART_DATA_WIDTH, 624
 - IS_UART_DIR, 624
 - IS_UART_ERROR, 624
 - IS_UART_FIFO_LEVEL, 625
 - IS_UART_FLAG, 625
 - IS_UART_GET_IT_SOURCE, 625
 - IS_UART_PARITY_BIT, 626
 - IS_UART_STOP_BIT, 626
 - UART_BaudRateDivConfig, 628
 - UART_Break, 629
 - UART_Cmd, 629
 - UART_DMABlkOnErrCmd, 630
 - UART_DMACmd, 630

- UART_DataWidth_5, 626
- UART_DataWidth_6, 626
- UART_DataWidth_7, 626
- UART_DataWidth_8, 626
- UART_DataWidth_TypeDef, 626
- UART_DeInit, 629
- UART_Dir_Rx, 627
- UART_Dir_Tx, 627
- UART_Dir_Typedef, 626
- UART_Error_Break, 627
- UART_Error_Frame, 627
- UART_Error_Overflow, 627
- UART_Error_Parity, 627
- UART_Error_Typedef, 627
- UART_ErrorStatus, 630
- UART_ErrorStatusClear, 630
- UART_FIFOLevel_1_2, 627
- UART_FIFOLevel_1_4, 627
- UART_FIFOLevel_1_8, 627
- UART_FIFOLevel_3_4, 627
- UART_FIFOLevel_7_8, 627
- UART_FIFOLevel_TypeDef, 627
- UART_Flag_Busy, 627
- UART_Flag_InvCTS, 627
- UART_Flag_InvDCD, 627
- UART_Flag_InvDSR, 627
- UART_Flag_InvRI, 627
- UART_Flag_RxFIFOEmpty, 627
- UART_Flag_RxFIFOFull, 627
- UART_Flag_TxFIFOEmpty, 627
- UART_Flag_TxFIFOFull, 627
- UART_Flag_Typedef, 627
- UART_FlagStatus, 632
- UART_ITCmd, 632
- UART_ITFIFOLevelConfig, 633
- UART_ITMaskedStatus, 633
- UART_ITRawStatus, 633
- UART_ITSource_ChangeCTS, 628
- UART_ITSource_ChangeDCD, 628
- UART_ITSource_ChangeDSR, 628
- UART_ITSource_ChangeRI, 628
- UART_ITSource_ErrorBreak, 628
- UART_ITSource_ErrorFrame, 628
- UART_ITSource_ErrorOverflow, 628
- UART_ITSource_ErrorParity, 628
- UART_ITSource_RecieveTimeout, 628
- UART_ITSource_RxFIFOLevel, 628
- UART_ITSource_TxFIFOLevel, 628
- UART_ITSource_Typedef, 627
- UART_ITStatusClear, 634
- UART_Init, 632
- UART_ModemConfig, 634
- UART_ModemStructInit, 634
- UART_ParityBit_Disable, 628
- UART_ParityBit_Even, 628
- UART_ParityBit_High, 628
- UART_ParityBit_Low, 628
- UART_ParityBit_Odd, 628
- UART_ParityBit_TypeDef, 628
- UART_RecieveData, 635
- UART_SendData, 635
- UART_StopBit_1, 628
- UART_StopBit_2, 628
- UART_StopBit_TypeDef, 628
- UART_StructInit, 635
- niietcm4_userflash.c, 636
 - USERFLASH_FullErase, 637
 - USERFLASH_ITCmd, 638
 - USERFLASH_Info_PageErase, 637
 - USERFLASH_Info_Read, 637
 - USERFLASH_Info_Write, 638
 - USERFLASH_Init, 638
 - USERFLASH_OperationStatus, 638
 - USERFLASH_OperationStatusClear, 639
 - USERFLASH_PageErase, 639
 - USERFLASH_Read, 639
 - USERFLASH_Write, 639
- niietcm4_userflash.h, 640
 - IS_USERFLASH_STATUS, 642
 - USERFLASH_FullErase, 643
 - USERFLASH_INFO_PAGE_SIZE_BYTES ← ES, 642
 - USERFLASH_INFO_PAGE_TOTAL, 642
 - USERFLASH_INFO_TOTAL_BYTES, 642
 - USERFLASH_ITCmd, 644
 - USERFLASH_Info_PageErase, 643
 - USERFLASH_Info_Read, 643
 - USERFLASH_Info_Write, 643
 - USERFLASH_Init, 644
 - USERFLASH_OperationStatus, 644
 - USERFLASH_OperationStatusClear, 644
 - USERFLASH_PAGE_SIZE_BYTES, 642
 - USERFLASH_PAGE_TOTAL, 642
 - USERFLASH_PageErase, 645
 - USERFLASH_Read, 645
 - USERFLASH_Status_Complete, 643
 - USERFLASH_Status_Error, 643
 - USERFLASH_Status_None, 643
 - USERFLASH_Status_TypeDef, 643
 - USERFLASH_TOTAL_BYTES, 642
 - USERFLASH_Write, 645
- niietcm4_watchdog.c, 645
 - WATCHDOG_Cmd, 647
 - WATCHDOG_GetCounter, 647
 - WATCHDOG_GetReload, 648
 - WATCHDOG_ITMaskedStatus, 648
 - WATCHDOG_ITRawStatus, 648
 - WATCHDOG_ITStatusClear, 648
 - WATCHDOG_Lock_Value, 647
 - WATCHDOG_LockCmd, 648
 - WATCHDOG_RstCmd, 648
 - WATCHDOG_SetReload, 649
 - WATCHDOG_Unlock_Value, 647
- niietcm4_watchdog.h, 649
 - WATCHDOG_Cmd, 650
 - WATCHDOG_GetCounter, 651

- WATCHDOG_GetReload, [651](#)
- WATCHDOG_ITMaskedStatus, [651](#)
- WATCHDOG_ITRawStatus, [651](#)
- WATCHDOG_ITStatusClear, [651](#)
- WATCHDOG_LockCmd, [651](#)
- WATCHDOG_RstCmd, [653](#)
- WATCHDOG_SetReload, [653](#)
- PRIVELGED
 - DMA_Protect_TypeDef, [387](#)
- PRM_DATA
 - DMA_ConfigData_TypeDef, [384](#)
- R_POWER
 - _CHANNEL_CFG_bits, [370](#)
- RCC, [313](#)
- RCC_ADCClk_0
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_1
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_10
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_11
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_2
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_3
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_4
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_5
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_6
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_7
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_8
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_9
 - Типы, [167](#)
 - niietcm4_rcc.h, [586](#)
- RCC_ADCClk_TypeDef
 - Типы, [167](#)
 - niietcm4_rcc.h, [585](#)
- RCC_ADCClkCmd
 - Приватные функции, [318](#)
 - Тактирование ADC, [183](#)
 - niietcm4_rcc.c, [571](#)
 - niietcm4_rcc.h, [589](#)
- RCC_ADCClkDivConfig
 - Приватные функции, [318](#)
 - Тактирование ADC, [183](#)
 - niietcm4_rcc.c, [572](#)
 - niietcm4_rcc.h, [589](#)
- RCC_CLK_CHANGE_TIMEOUT
 - Константы, [162](#)
 - niietcm4_rcc.h, [585](#)
- RCC_CLK_PLL_STABLE_TIMEOUT
 - Константы, [162](#)
 - niietcm4_rcc.h, [585](#)
- RCC_PLL_CTRL_Reset_Value
 - Начальные значения регистров, [316](#)
 - niietcm4_rcc.c, [571](#)
- RCC_PLL_NF_Reset_Value
 - Начальные значения регистров, [316](#)
 - niietcm4_rcc.c, [571](#)
- RCC_PLL_NR_Reset_Value
 - Начальные значения регистров, [316](#)
 - niietcm4_rcc.c, [571](#)
- RCC_PLL_OD_Reset_Value
 - Начальные значения регистров, [316](#)
 - niietcm4_rcc.c, [571](#)
- RCC_PLLAutoConfig
 - Конфигурация PLL, [172](#)
 - Приватные функции, [319](#)
 - niietcm4_rcc.c, [573](#)
 - niietcm4_rcc.h, [591](#)
- RCC_PLLDeInit
 - Конфигурация PLL, [172](#)
 - Приватные функции, [320](#)
 - niietcm4_rcc.c, [574](#)
 - niietcm4_rcc.h, [591](#)
- RCC_PLLDiv
 - RCC_PLLInit_TypeDef, [390](#)
- RCC_PLLInit
 - Конфигурация PLL, [173](#)
 - Приватные функции, [320](#)
 - niietcm4_rcc.c, [574](#)
 - niietcm4_rcc.h, [591](#)
- RCC_PLLInit_TypeDef, [390](#)
 - RCC_PLLDiv, [390](#)
 - RCC_PLLNF, [390](#)
 - RCC_PLLNO, [390](#)
 - RCC_PLLNR, [391](#)
 - RCC_PLLRef, [391](#)
- RCC_PLLNF
 - RCC_PLLInit_TypeDef, [390](#)
- RCC_PLLNO
 - RCC_PLLInit_TypeDef, [390](#)
- RCC_PLLNO_Disable
 - Типы, [169](#)
 - niietcm4_rcc.h, [587](#)
- RCC_PLLNO_Div2
 - Типы, [169](#)
 - niietcm4_rcc.h, [587](#)
- RCC_PLLNO_Div4

- Типы, [169](#)
- [niietcm4_rcc.h](#), [587](#)
- RCC_PLLNO_TypeDef
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PLLNR
 - RCC_PLLInit_TypeDef, [391](#)
- RCC_PLLPowerDownCmd
 - Конфигурация PLL, [173](#)
 - Приватные функции, [321](#)
 - [niietcm4_rcc.c](#), [575](#)
 - [niietcm4_rcc.h](#), [592](#)
- RCC_PLLRef
 - RCC_PLLInit_TypeDef, [391](#)
- RCC_PLLRef_ETH_25MHz
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [588](#)
- RCC_PLLRef_TypeDef
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PLLRef_USB_60MHz
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [588](#)
- RCC_PLLRef_USB_CLK
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [588](#)
- RCC_PLLRef_XI_OSC
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [588](#)
- RCC_PLLStructInit
 - Конфигурация PLL, [174](#)
 - Приватные функции, [321](#)
 - [niietcm4_rcc.c](#), [575](#)
 - [niietcm4_rcc.h](#), [593](#)
- RCC_PeriphClk_ADC
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_CMP
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_I2C0
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_I2C1
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM0
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM1
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM2
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM3
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM4
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM5
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM6
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM7
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_PWM8
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_QEP0
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_QEP1
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_TypeDef
 - Типы, [167](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClk_WD
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [586](#)
- RCC_PeriphClkCmd
 - Приватные функции, [318](#)
 - Управление тактированием, [175](#)
 - [niietcm4_rcc.c](#), [572](#)
 - [niietcm4_rcc.h](#), [590](#)
- RCC_PeriphRst_CAP0
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_CAP1
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_CAP2
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_CAP3
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_CAP4
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_CAP5
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_CMP
 - Типы, [169](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_ETH
 - Типы, [168](#)
 - [niietcm4_rcc.h](#), [587](#)
- RCC_PeriphRst_I2C0
 - Типы, [168](#)

- niietcm4_rcc.h, 586
- RCC_PeriphRst_I2C1
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM0
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM1
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM2
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM3
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM4
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM5
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM6
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM7
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_PWM8
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_QEP0
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_QEP1
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_SPI0
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_SPI1
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_SPI2
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_SPI3
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_Timer0
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_Timer1
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_Timer2
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_TypeDef
 - Типы, 168
- niietcm4_rcc.h, 586
- RCC_PeriphRst_UART0
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_UART1
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_UART2
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_UART3
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_USB
 - Типы, 168
- niietcm4_rcc.h, 587
- RCC_PeriphRst_WD
 - Типы, 168
- niietcm4_rcc.h, 586
- RCC_PeriphRstCmd
 - Приватные функции, 319
 - Управление сбросом, 184
- niietcm4_rcc.c, 573
 - niietcm4_rcc.h, 590
- RCC_SPIClk_SYSClk
 - Типы, 169
- niietcm4_rcc.h, 588
- RCC_SPIClk_TypeDef
 - Типы, 169
- niietcm4_rcc.h, 588
- RCC_SPIClk_USB_60MHz
 - Типы, 169
- niietcm4_rcc.h, 588
- RCC_SPIClk_USB_CLK
 - Типы, 169
- niietcm4_rcc.h, 588
- RCC_SPIClk_XI_OSC
 - Типы, 169
- niietcm4_rcc.h, 588
- RCC_SPIClkCmd
 - Приватные функции, 321
 - Тактирование SPI, 181
- niietcm4_rcc.c, 575
 - niietcm4_rcc.h, 593
- RCC_SPIClkDivConfig
 - Приватные функции, 322
 - Тактирование SPI, 181
- niietcm4_rcc.c, 576
 - niietcm4_rcc.h, 593
- RCC_SPIClkSel
 - Приватные функции, 322
 - Тактирование SPI, 181
- niietcm4_rcc.c, 576
 - niietcm4_rcc.h, 594
- RCC_SysClk_CPE_Sel
 - Типы, 170

- niietcm4_rcc.h, 588
- RCC_SysClk_ETH25MHz
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClk_PLL
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClk_PLLDIV
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClk_POR
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClk_TypeDef
 - Типы, 169
 - niietcm4_rcc.h, 588
- RCC_SysClk_USB60MHz
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClk_USB_CLK
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClk_XI_OSC
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_SysClkDiv2Out
 - Функции, 171
 - Приватные функции, 322
 - niietcm4_rcc.c, 576
 - niietcm4_rcc.h, 594
- RCC_SysClkSel
 - Приватные функции, 323
 - Управление тактированием, 175
 - niietcm4_rcc.c, 577
 - niietcm4_rcc.h, 594
- RCC_SysClkStatus
 - Приватные функции, 323
 - Управление тактированием, 176
 - niietcm4_rcc.c, 577
 - niietcm4_rcc.h, 595
- RCC_UARTClk_SYSCLK
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_UARTClk_TypeDef
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_UARTClk_USB_60MHz
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_UARTClk_USB_CLK
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_UARTClk_XI_OSC
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_UARTClkCmd
 - Приватные функции, 323
 - Тактирование UART, 178
- niietcm4_rcc.c, 577
 - niietcm4_rcc.h, 595
- RCC_UARTClkDivConfig
 - Приватные функции, 324
 - Тактирование UART, 178
 - niietcm4_rcc.c, 578
 - niietcm4_rcc.h, 595
- RCC_UARTClkSel
 - Приватные функции, 324
 - Тактирование UART, 178
 - niietcm4_rcc.c, 578
 - niietcm4_rcc.h, 595
- RCC_USBClk_TypeDef
 - Типы, 170
 - niietcm4_rcc.h, 588
- RCC_USBClk_USB_CLK
 - Типы, 170
 - niietcm4_rcc.h, 589
- RCC_USBClk_XI_OSC
 - Типы, 170
 - niietcm4_rcc.h, 589
- RCC_USBClkCmd
 - Приватные функции, 324
 - Тактирование USB, 177
 - niietcm4_rcc.c, 578
 - niietcm4_rcc.h, 596
- RCC_USBClkConfig
 - Приватные функции, 325
 - Тактирование USB, 177
 - niietcm4_rcc.c, 579
 - niietcm4_rcc.h, 596
- RCC_USBFreq_12MHz
 - Типы, 170
 - niietcm4_rcc.h, 589
- RCC_USBFreq_24MHz
 - Типы, 170
 - niietcm4_rcc.h, 589
- RCC_USBFreq_TypeDef
 - Типы, 170
 - niietcm4_rcc.h, 589
- RCC_WaitClkChange
 - Приватные функции, 325
 - niietcm4_rcc.c, 579
- RESERVED0
 - DMA_ConfigData_TypeDef, 384
- RESERVED1
 - DMA_ConfigData_TypeDef, 384
- RTC, 326
 - RTC_Date_TypeDef, 391
 - RTC_Day, 391
 - RTC_Month, 391
 - RTC_Weekday, 392
 - RTC_Year, 392
- RTC_Day
 - RTC_Date_TypeDef, 391
- RTC_Format_BCD
 - Типы, 186
 - niietcm4_rtc.h, 600

- RTC_Format_BIN
 - Типы, [186](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Format_TypeDef
 - Типы, [186](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Hour
 - RTC_Time_TypeDef, [392](#)
- RTC_Minute
 - RTC_Time_TypeDef, [392](#)
- RTC_Month
 - RTC_Date_TypeDef, [391](#)
- RTC_Month_April
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_August
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_December
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_February
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_January
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_July
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_June
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_March
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_May
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_November
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_October
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_September
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Month_TypeDef
 - Типы, [186](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Psecond
 - RTC_Time_TypeDef, [393](#)
- RTC_Second
 - RTC_Time_TypeDef, [393](#)
- RTC_Time_TypeDef, [392](#)
 - RTC_Hour, [392](#)
 - RTC_Minute, [392](#)
 - RTC_Psecond, [393](#)
 - RTC_Second, [393](#)
- RTC_Weekday
 - RTC_Date_TypeDef, [392](#)
- RTC_Weekday_Friday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Weekday_Monday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Weekday_Saturday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Weekday_Sunday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Weekday_Thursday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Weekday_Tuesday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Weekday_TypeDef
 - Типы, [187](#)
 - niietcm4_rtc.h, [600](#)
- RTC_Weekday_Wednesday
 - Типы, [187](#)
 - niietcm4_rtc.h, [601](#)
- RTC_Year
 - RTC_Date_TypeDef, [392](#)
- SRC_DATA_END
 - DMA_Channel_TypeDef, [380](#)
- SRC_INC
 - _CHANNEL_CFG_bits, [371](#)
- SRC_PROT_BUFFERABLE
 - _CHANNEL_CFG_bits, [371](#)
- SRC_PROT_CACHEABLE
 - _CHANNEL_CFG_bits, [371](#)
- SRC_PROT_PRIVILEGED
 - _CHANNEL_CFG_bits, [371](#)
- SRC_SIZE
 - _CHANNEL_CFG_bits, [371](#)
- TIMER, [329](#)
 - TIMER_Cmd
 - Конфигурация, [192](#)
 - Приватные функции, [332](#)
 - niietcm4_timer.c, [602](#)
 - niietcm4_timer.h, [608](#)
 - TIMER_ExtInput_CountClk
 - Типы, [189](#)
 - niietcm4_timer.h, [607](#)
 - TIMER_ExtInput_CountEn
 - Типы, [189](#)
 - niietcm4_timer.h, [607](#)
 - TIMER_ExtInput_Disable
 - Типы, [189](#)
 - niietcm4_timer.h, [607](#)

- TIMER_ExtInput_TypeDef
 - Типы, [189](#)
 - niietcm4_timer.h, [607](#)
- TIMER_ExtInputConfig
 - Конфигурация, [192](#)
 - Приватные функции, [332](#)
 - niietcm4_timer.c, [602](#)
 - niietcm4_timer.h, [609](#)
- TIMER_FreqConfig
 - Конфигурация, [193](#)
 - Приватные функции, [333](#)
 - niietcm4_timer.c, [603](#)
 - niietcm4_timer.h, [609](#)
- TIMER_GetCounter
 - Конфигурация, [193](#)
 - Приватные функции, [333](#)
 - niietcm4_timer.c, [603](#)
 - niietcm4_timer.h, [609](#)
- TIMER_GetReload
 - Конфигурация, [193](#)
 - Приватные функции, [333](#)
 - niietcm4_timer.c, [603](#)
 - niietcm4_timer.h, [610](#)
- TIMER_ITCmd
 - Прерывания, [196](#)
 - Приватные функции, [334](#)
 - niietcm4_timer.c, [604](#)
 - niietcm4_timer.h, [610](#)
- TIMER_ITStatus
 - Прерывания, [196](#)
 - Приватные функции, [334](#)
 - niietcm4_timer.c, [604](#)
 - niietcm4_timer.h, [610](#)
- TIMER_ITStatusClear
 - Прерывания, [196](#)
 - Приватные функции, [334](#)
 - niietcm4_timer.c, [604](#)
 - niietcm4_timer.h, [610](#)
- TIMER_PeriodConfig
 - Конфигурация, [194](#)
 - Приватные функции, [334](#)
 - niietcm4_timer.c, [604](#)
 - niietcm4_timer.h, [611](#)
- TIMER_SetCounter
 - Конфигурация, [194](#)
 - Приватные функции, [335](#)
 - niietcm4_timer.c, [605](#)
 - niietcm4_timer.h, [611](#)
- TIMER_SetReload
 - Конфигурация, [194](#)
 - Приватные функции, [335](#)
 - niietcm4_timer.c, [605](#)
 - niietcm4_timer.h, [611](#)
- UART, [336](#)
- UART_BaudRate
 - UART_Init_TypeDef, [393](#)
- UART_BaudRateDivConfig
 - Функции, [205](#)
 - Приватные функции, [340](#)
 - niietcm4_uart.c, [614](#)
 - niietcm4_uart.h, [628](#)
- UART_Break
 - Функции, [205](#)
 - Приватные функции, [340](#)
 - niietcm4_uart.c, [614](#)
 - niietcm4_uart.h, [629](#)
- UART_CTSEn
 - UART_ModemInit_TypeDef, [395](#)
- UART_ClkFreq
 - UART_Init_TypeDef, [394](#)
- UART_Cmd
 - Функции, [206](#)
 - Приватные функции, [340](#)
 - niietcm4_uart.c, [614](#)
 - niietcm4_uart.h, [629](#)
- UART_DMABlkOnErrCmd
 - Настройка DMA, [217](#)
 - Приватные функции, [341](#)
 - niietcm4_uart.c, [616](#)
 - niietcm4_uart.h, [630](#)
- UART_DMACmd
 - Настройка DMA, [217](#)
 - Приватные функции, [341](#)
 - niietcm4_uart.c, [616](#)
 - niietcm4_uart.h, [630](#)
- UART_DataWidth
 - UART_Init_TypeDef, [394](#)
- UART_DataWidth_5
 - Типы, [201](#)
 - niietcm4_uart.h, [626](#)
- UART_DataWidth_6
 - Типы, [201](#)
 - niietcm4_uart.h, [626](#)
- UART_DataWidth_7
 - Типы, [201](#)
 - niietcm4_uart.h, [626](#)
- UART_DataWidth_8
 - Типы, [201](#)
 - niietcm4_uart.h, [626](#)
- UART_DataWidth_TypeDef
 - Типы, [201](#)
 - niietcm4_uart.h, [626](#)
- UART_DeInit
 - Инициализация и деинициализация, [207](#)
 - Приватные функции, [341](#)
 - niietcm4_uart.c, [614](#)
 - niietcm4_uart.h, [629](#)
- UART_Dir_Rx
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Dir_Tx
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Dir_Typedef
 - Типы, [201](#)
 - niietcm4_uart.h, [626](#)

- UART_Error_Break
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Error_Frame
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Error_Overflow
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Error_Parity
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Error_Typedef
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_ErrorStatus
 - Прием и передача, [210](#)
 - Приватные функции, [341](#)
 - niietcm4_uart.c, [616](#)
 - niietcm4_uart.h, [630](#)
- UART_ErrorStatusClear
 - Прием и передача, [210](#)
 - Приватные функции, [342](#)
 - niietcm4_uart.c, [617](#)
 - niietcm4_uart.h, [630](#)
- UART_FIFOEn
 - UART_Init_TypeDef, [394](#)
- UART_FIFOLevel_1_2
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FIFOLevel_1_4
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FIFOLevel_1_8
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FIFOLevel_3_4
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FIFOLevel_7_8
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FIFOLevel_TypeDef
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FIFOLevelRx
 - UART_Init_TypeDef, [394](#)
- UART_FIFOLevelTx
 - UART_Init_TypeDef, [394](#)
- UART_Flag_Busy
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_InvCTS
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_InvDCD
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_InvDSR
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_InvRI
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_RxFIFOEmpty
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_RxFIFOFull
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_TxFIFOEmpty
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_TxFIFOFull
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_Flag_Typedef
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_FlagStatus
 - Прием и передача, [210](#)
 - Приватные функции, [342](#)
 - niietcm4_uart.c, [617](#)
 - niietcm4_uart.h, [632](#)
- UART_ITCmd
 - Прерывания, [214](#)
 - Приватные функции, [343](#)
 - niietcm4_uart.c, [618](#)
 - niietcm4_uart.h, [632](#)
- UART_ITFIFOLevelConfig
 - Прерывания, [214](#)
 - Приватные функции, [343](#)
 - niietcm4_uart.c, [618](#)
 - niietcm4_uart.h, [633](#)
- UART_ITMaskedStatus
 - Прерывания, [215](#)
 - Приватные функции, [343](#)
 - niietcm4_uart.c, [618](#)
 - niietcm4_uart.h, [633](#)
- UART_ITRawStatus
 - Прерывания, [215](#)
 - Приватные функции, [345](#)
 - niietcm4_uart.c, [619](#)
 - niietcm4_uart.h, [633](#)
- UART_ITSource_ChangeCTS
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_ChangeDCD
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_ChangeDSR
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_ChangeRI
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)

- UART_ITSource_ErrorBreak
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_ErrorFrame
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_ErrorOverflow
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_ErrorParity
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_RecieveTimeout
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_RxFIFOLevel
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_TxFIFOLevel
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ITSource_Typedef
 - Типы, [202](#)
 - niietcm4_uart.h, [627](#)
- UART_ITStatusClear
 - Прерывания, [215](#)
 - Приватные функции, [345](#)
 - niietcm4_uart.c, [619](#)
 - niietcm4_uart.h, [634](#)
- UART_Init
 - Инициализация и деинициализация, [207](#)
 - Приватные функции, [342](#)
 - niietcm4_uart.c, [617](#)
 - niietcm4_uart.h, [632](#)
- UART_Init_TypeDef, [393](#)
 - UART_BaudRate, [393](#)
 - UART_ClkFreq, [394](#)
 - UART_DataWidth, [394](#)
 - UART_FIFOEn, [394](#)
 - UART_FIFOLevelRx, [394](#)
 - UART_FIFOLevelTx, [394](#)
 - UART_ParityBit, [394](#)
 - UART_RxEn, [394](#)
 - UART_StopBit, [395](#)
 - UART_TxEn, [395](#)
- UART_InvDTR
 - UART_ModemInit_TypeDef, [395](#)
- UART_InvRTS
 - UART_ModemInit_TypeDef, [396](#)
- UART_ModemConfig
 - Приватные функции, [345](#)
 - Режим модема, [213](#)
 - niietcm4_uart.c, [619](#)
 - niietcm4_uart.h, [634](#)
- UART_ModemInit_TypeDef, [395](#)
 - UART_CTSEn, [395](#)
 - UART_InvDTR, [395](#)
 - UART_InvRTS, [396](#)
 - UART_RTSEn, [396](#)
- UART_ModemStructInit
 - Приватные функции, [346](#)
 - Режим модема, [213](#)
 - niietcm4_uart.c, [620](#)
 - niietcm4_uart.h, [634](#)
- UART_ParityBit
 - UART_Init_TypeDef, [394](#)
- UART_ParityBit_Disable
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ParityBit_Even
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ParityBit_High
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ParityBit_Low
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ParityBit_Odd
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_ParityBit_TypeDef
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_RTSEn
 - UART_ModemInit_TypeDef, [396](#)
- UART_RecieveData
 - Прием и передача, [212](#)
 - Приватные функции, [346](#)
 - niietcm4_uart.c, [620](#)
 - niietcm4_uart.h, [635](#)
- UART_RxEn
 - UART_Init_TypeDef, [394](#)
- UART_SendData
 - Прием и передача, [212](#)
 - Приватные функции, [346](#)
 - niietcm4_uart.c, [620](#)
 - niietcm4_uart.h, [635](#)
- UART_StopBit
 - UART_Init_TypeDef, [395](#)
- UART_StopBit_1
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_StopBit_2
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_StopBit_TypeDef
 - Типы, [203](#)
 - niietcm4_uart.h, [628](#)
- UART_StructInit
 - Инициализация и деинициализация, [207](#)
 - Приватные функции, [346](#)
 - niietcm4_uart.c, [620](#)
 - niietcm4_uart.h, [635](#)
- UART_TxEn
 - UART_Init_TypeDef, [395](#)

- USERFLASH, 349
- USERFLASH_FullErase
 - Основная область флеш, 224
 - Приватные функции, 352
 - niietcm4_userflash.c, 637
 - niietcm4_userflash.h, 643
- USERFLASH_INFO_PAGE_SIZE_BYTES
 - Информационная область флеш, 220
 - niietcm4_userflash.h, 642
- USERFLASH_INFO_PAGE_TOTAL
 - Информационная область флеш, 220
 - niietcm4_userflash.h, 642
- USERFLASH_INFO_TOTAL_BYTES
 - Информационная область флеш, 220
 - niietcm4_userflash.h, 642
- USERFLASH_ITCmd
 - Функции, 222
 - Приватные функции, 353
 - niietcm4_userflash.c, 638
 - niietcm4_userflash.h, 644
- USERFLASH_Info_PageErase
 - Информационная область флеш, 226
 - Приватные функции, 352
 - niietcm4_userflash.c, 637
 - niietcm4_userflash.h, 643
- USERFLASH_Info_Read
 - Информационная область флеш, 226
 - Приватные функции, 353
 - niietcm4_userflash.c, 637
 - niietcm4_userflash.h, 643
- USERFLASH_Info_Write
 - Информационная область флеш, 226
 - Приватные функции, 353
 - niietcm4_userflash.c, 638
 - niietcm4_userflash.h, 643
- USERFLASH_Init
 - Функции, 222
 - Приватные функции, 353
 - niietcm4_userflash.c, 638
 - niietcm4_userflash.h, 644
- USERFLASH_OperationStatus
 - Функции, 222
 - Приватные функции, 354
 - niietcm4_userflash.c, 638
 - niietcm4_userflash.h, 644
- USERFLASH_OperationStatusClear
 - Функции, 223
 - Приватные функции, 354
 - niietcm4_userflash.c, 639
 - niietcm4_userflash.h, 644
- USERFLASH_PAGE_SIZE_BYTES
 - Основная область флеш, 219
 - niietcm4_userflash.h, 642
- USERFLASH_PAGE_TOTAL
 - Основная область флеш, 219
 - niietcm4_userflash.h, 642
- USERFLASH_PageErase
 - Основная область флеш, 224
- Приватные функции, 354
 - niietcm4_userflash.c, 639
 - niietcm4_userflash.h, 645
- USERFLASH_Read
 - Основная область флеш, 224
 - Приватные функции, 354
 - niietcm4_userflash.c, 639
 - niietcm4_userflash.h, 645
- USERFLASH_Status_Complete
 - Типы, 221
 - niietcm4_userflash.h, 643
- USERFLASH_Status_Error
 - Типы, 221
 - niietcm4_userflash.h, 643
- USERFLASH_Status_None
 - Типы, 221
 - niietcm4_userflash.h, 643
- USERFLASH_Status_TypeDef
 - Типы, 221
 - niietcm4_userflash.h, 643
- USERFLASH_TOTAL_BYTES
 - Основная область флеш, 219
 - niietcm4_userflash.h, 642
- USERFLASH_Write
 - Основная область флеш, 225
 - Приватные функции, 355
 - niietcm4_userflash.c, 639
 - niietcm4_userflash.h, 645
- WATCHDOG, 356
- WATCHDOG_Cmd
 - Конфигурация, 231
 - Приватные функции, 359
 - niietcm4_watchdog.c, 647
 - niietcm4_watchdog.h, 650
- WATCHDOG_GetCounter
 - Конфигурация, 231
 - Приватные функции, 359
 - niietcm4_watchdog.c, 647
 - niietcm4_watchdog.h, 651
- WATCHDOG_GetReload
 - Конфигурация, 231
 - Приватные функции, 359
 - niietcm4_watchdog.c, 648
 - niietcm4_watchdog.h, 651
- WATCHDOG_ITMaskedStatus
 - Прерывания, 233
 - Приватные функции, 360
 - niietcm4_watchdog.c, 648
 - niietcm4_watchdog.h, 651
- WATCHDOG_ITRawStatus
 - Прерывания, 233
 - Приватные функции, 360
 - niietcm4_watchdog.c, 648
 - niietcm4_watchdog.h, 651
- WATCHDOG_ITStatusClear
 - Прерывания, 233
 - Приватные функции, 360
 - niietcm4_watchdog.c, 648

- [niietcm4_watchdog.h](#), [651](#)
 - [WATCHDOG_Lock_Value](#)
 - Приватные константы, [358](#)
 - [niietcm4_watchdog.c](#), [647](#)
 - [WATCHDOG_LockCmd](#)
 - Конфигурация, [232](#)
 - Приватные функции, [360](#)
 - [niietcm4_watchdog.c](#), [648](#)
 - [niietcm4_watchdog.h](#), [651](#)
 - [WATCHDOG_RstCmd](#)
 - Конфигурация, [232](#)
 - Приватные функции, [360](#)
 - [niietcm4_watchdog.c](#), [648](#)
 - [niietcm4_watchdog.h](#), [653](#)
 - [WATCHDOG_SetReload](#)
 - Конфигурация, [232](#)
 - Приватные функции, [361](#)
 - [niietcm4_watchdog.c](#), [649](#)
 - [niietcm4_watchdog.h](#), [653](#)
 - [WATCHDOG_Unlock_Value](#)
 - Приватные константы, [358](#)
 - [niietcm4_watchdog.c](#), [647](#)