

НИЕТСМ4 PD  
v0.9.0

Создано системой Doxygen 1.8.10

Вт 2 Авг 2016 13:51:06



# 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	76

6.24.2.7	CAP_ITSource_TimerEqCompare . . . . .	77
6.24.2.8	CAP_ITSource_TimerEqPeriod . . . . .	77
6.24.2.9	CAP_ITSource_TimerOvf . . . . .	77
6.25	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 номеру	102
6.33.1	Подробное описание	102
6.33.2	Макросы	102
6.33.2.1	DMA_Channel_0	102
6.33.2.2	DMA_Channel_1	102
6.33.2.3	DMA_Channel_10	102
6.33.2.4	DMA_Channel_11	102
6.33.2.5	DMA_Channel_12	103
6.33.2.6	DMA_Channel_13	103
6.33.2.7	DMA_Channel_14	103
6.33.2.8	DMA_Channel_15	103
6.33.2.9	DMA_Channel_16	103
6.33.2.10	DMA_Channel_17	103
6.33.2.11	DMA_Channel_18	103
6.33.2.12	DMA_Channel_19	103
6.33.2.13	DMA_Channel_2	103
6.33.2.14	DMA_Channel_20	103
6.33.2.15	DMA_Channel_21	104
6.33.2.16	DMA_Channel_22	104
6.33.2.17	DMA_Channel_23	104
6.33.2.18	DMA_Channel_3	104
6.33.2.19	DMA_Channel_4	104
6.33.2.20	DMA_Channel_5	104
6.33.2.21	DMA_Channel_6	104
6.33.2.22	DMA_Channel_7	104
6.33.2.23	DMA_Channel_8	104
6.33.2.24	DMA_Channel_9	105
6.34	Маски каналов по имени	106

6.34.1	Подробное описание . . . . .	106
6.34.2	Макросы . . . . .	106
6.34.2.1	DMA_Channel_ADCSEQ0 . . . . .	106
6.34.2.2	DMA_Channel_ADCSEQ1 . . . . .	106
6.34.2.3	DMA_Channel_ADCSEQ2 . . . . .	106
6.34.2.4	DMA_Channel_ADCSEQ3 . . . . .	106
6.34.2.5	DMA_Channel_ADCSEQ4 . . . . .	107
6.34.2.6	DMA_Channel_ADCSEQ5 . . . . .	107
6.34.2.7	DMA_Channel_ADCSEQ6 . . . . .	107
6.34.2.8	DMA_Channel_ADCSEQ7 . . . . .	107
6.34.2.9	DMA_Channel_SPI0_RX . . . . .	107
6.34.2.10	DMA_Channel_SPI0_TX . . . . .	107
6.34.2.11	DMA_Channel_SPI1_RX . . . . .	107
6.34.2.12	DMA_Channel_SPI1_TX . . . . .	107
6.34.2.13	DMA_Channel_SPI2_RX . . . . .	107
6.34.2.14	DMA_Channel_SPI2_TX . . . . .	107
6.34.2.15	DMA_Channel_SPI3_RX . . . . .	108
6.34.2.16	DMA_Channel_SPI3_TX . . . . .	108
6.34.2.17	DMA_Channel_UART0_RX . . . . .	108
6.34.2.18	DMA_Channel_UART0_TX . . . . .	108
6.34.2.19	DMA_Channel_UART1_RX . . . . .	108
6.34.2.20	DMA_Channel_UART1_TX . . . . .	108
6.34.2.21	DMA_Channel_UART2_RX . . . . .	108
6.34.2.22	DMA_Channel_UART2_TX . . . . .	108
6.34.2.23	DMA_Channel_UART3_RX . . . . .	108
6.34.2.24	DMA_Channel_UART3_TX . . . . .	109
6.35	Типы . . . . .	110
6.35.1	Подробное описание . . . . .	111
6.35.2	Макросы . . . . .	111
6.35.2.1	IS_DMA_ARBITRATION_RATE . . . . .	111
6.35.2.2	IS_DMA_DATA_INC . . . . .	111
6.35.2.3	IS_DMA_DATA_SIZE . . . . .	112
6.35.2.4	IS_DMA_MODE . . . . .	112
6.35.2.5	IS_DMA_STATE . . . . .	112
6.35.3	Перечисления . . . . .	112
6.35.3.1	DMA_ArbitrationRate_TypeDef . . . . .	112
6.35.3.2	DMA_DataInc_TypeDef . . . . .	113
6.35.3.3	DMA_DataSize_TypeDef . . . . .	113
6.35.3.4	DMA_Mode_TypeDef . . . . .	113
6.35.3.5	DMA_State_TypeDef . . . . .	114

6.36	Функции	115
6.36.1	Подробное описание	115
6.37	Инициализация каналов DMA	116
6.37.1	Подробное описание	116
6.37.2	Функции	116
6.37.2.1	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	116
6.37.2.2	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	116
6.37.2.3	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	117
6.38	Инициализация контроллера DMA	118
6.38.1	Подробное описание	118
6.38.2	Функции	118
6.38.2.1	DMA_DeInit()	118
6.38.2.2	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	118
6.38.2.3	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	118
6.39	Конфигурация контроллера DMA	120
6.39.1	Подробное описание	120
6.39.2	Функции	120
6.39.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	120
6.39.2.2	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	120
6.39.2.3	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	121
6.39.2.4	DMA_MasterEnableCmd(FunctionalState State)	121
6.39.2.5	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	121
6.39.2.6	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	122
6.39.2.7	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	122
6.39.2.8	DMA_SWRequestCmd(uint32_t DMA_Channel)	122
6.39.2.9	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	123
6.40	Статусная информация	124
6.40.1	Подробное описание	124
6.40.2	Функции	124
6.40.2.1	DMA_ClearErrorStatus()	124
6.40.2.2	DMA_ErrorStatus()	124
6.40.2.3	DMA_MasterEnableStatus()	124
6.40.2.4	DMA_StateStatus()	124
6.40.2.5	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	125
6.41	Константы	126
6.41.1	Подробное описание	126
6.42	Маски адреса	127
6.42.1	Подробное описание	127

6.42.2	Макросы	127
6.42.2.1	EXTMEM_CEMask_Addr_11	127
6.42.2.2	EXTMEM_CEMask_Addr_11_19	127
6.42.2.3	EXTMEM_CEMask_Addr_12	127
6.42.2.4	EXTMEM_CEMask_Addr_13	127
6.42.2.5	EXTMEM_CEMask_Addr_14	127
6.42.2.6	EXTMEM_CEMask_Addr_15	127
6.42.2.7	EXTMEM_CEMask_Addr_16	128
6.42.2.8	EXTMEM_CEMask_Addr_17	128
6.42.2.9	EXTMEM_CEMask_Addr_18	128
6.42.2.10	EXTMEM_CEMask_Addr_19	128
6.43	Типы	129
6.43.1	Подробное описание	129
6.43.2	Макросы	129
6.43.2.1	IS_EXTMEM_READ_WAITSTATE	129
6.43.2.2	IS_EXTMEM_RW_WAITSTATE	130
6.43.2.3	IS_EXTMEM_WIDTH	130
6.43.2.4	IS_EXTMEM_WRITE_WAITSTATE	130
6.43.3	Перечисления	131
6.43.3.1	EXTMEM_ReadWaitState_TypeDef	131
6.43.3.2	EXTMEM_RWWaitState_TypeDef	131
6.43.3.3	EXTMEM_Width_TypeDef	131
6.43.3.4	EXTMEM_WriteWaitState_TypeDef	132
6.44	Функции	133
6.44.1	Подробное описание	133
6.44.2	Функции	133
6.44.2.1	EXTMEM_DeInit()	133
6.44.2.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	133
6.44.2.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	133
6.45	Типы	135
6.45.1	Подробное описание	136
6.45.2	Макросы	136
6.45.2.1	IS_GPIO_ALT_FUNC	136
6.45.2.2	IS_GPIO_DIR	136
6.45.2.3	IS_GPIO_INT_POL	136
6.45.2.4	IS_GPIO_INT_TYPE	137
6.45.2.5	IS_GPIO_LOAD	137
6.45.2.6	IS_GPIO_MODE	137
6.45.2.7	IS_GPIO_OUT	137
6.45.2.8	IS_GPIO_OUT_MODE	137

6.45.2.9	IS_GPIO_PULLUP	138
6.45.2.10	IS_GPIO_QUAL	138
6.45.2.11	IS_GPIO_QUAL_MODE	138
6.45.2.12	IS_GPIO_SYNC	138
6.45.3	Перечисления	138
6.45.3.1	BitAction	138
6.45.3.2	GPIO_AltFunc_TypeDef	139
6.45.3.3	GPIO_Dir_TypeDef	139
6.45.3.4	GPIO_IntPol_TypeDef	139
6.45.3.5	GPIO_IntType_TypeDef	139
6.45.3.6	GPIO_Load_TypeDef	139
6.45.3.7	GPIO_Mode_TypeDef	140
6.45.3.8	GPIO_Out_TypeDef	140
6.45.3.9	GPIO_OutMode_TypeDef	140
6.45.3.10	GPIO_PullUp_TypeDef	140
6.45.3.11	GPIO_Qual_TypeDef	140
6.45.3.12	GPIO_QualMode_TypeDef	141
6.45.3.13	GPIO_Sync_TypeDef	141
6.46	Константы	142
6.46.1	Подробное описание	142
6.47	Маски пинов	143
6.47.1	Подробное описание	143
6.47.2	Макросы	143
6.47.2.1	GPIO_Pin_0	143
6.47.2.2	GPIO_Pin_0_3	143
6.47.2.3	GPIO_Pin_0_7	143
6.47.2.4	GPIO_Pin_1	144
6.47.2.5	GPIO_Pin_10	144
6.47.2.6	GPIO_Pin_11	144
6.47.2.7	GPIO_Pin_12	144
6.47.2.8	GPIO_Pin_12_15	144
6.47.2.9	GPIO_Pin_13	144
6.47.2.10	GPIO_Pin_14	144
6.47.2.11	GPIO_Pin_15	144
6.47.2.12	GPIO_Pin_2	144
6.47.2.13	GPIO_Pin_3	144
6.47.2.14	GPIO_Pin_4	145
6.47.2.15	GPIO_Pin_4_7	145
6.47.2.16	GPIO_Pin_5	145
6.47.2.17	GPIO_Pin_6	145



6.47.2.18	GPIO_Pin_7	145
6.47.2.19	GPIO_Pin_8	145
6.47.2.20	GPIO_Pin_8_11	145
6.47.2.21	GPIO_Pin_8_15	145
6.47.2.22	GPIO_Pin_9	145
6.47.2.23	GPIO_Pin_All	146
6.47.2.24	IS_GET_GPIO_PIN	146
6.48	Функции	147
6.48.1	Подробное описание	147
6.49	Инициализация и деинициализация	148
6.49.1	Подробное описание	148
6.49.2	Функции	148
6.49.2.1	GPIO_AltFuncConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc)	148
6.49.2.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	148
6.49.2.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	149
6.49.2.4	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	149
6.50	Чтение и запись	150
6.50.1	Подробное описание	150
6.51	Чтение	151
6.51.1	Подробное описание	151
6.51.2	Функции	151
6.51.2.1	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	151
6.51.2.2	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	151
6.51.2.3	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	151
6.52	Запись	153
6.52.1	Подробное описание	153
6.52.2	Функции	153
6.52.2.1	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	153
6.52.2.2	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	153
6.52.2.3	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	153
6.53	Битовые операции	155
6.53.1	Подробное описание	155
6.53.2	Функции	155
6.53.2.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	155
6.53.2.2	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	155
6.53.2.3	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	155
6.54	Фильтрация	157

6.54.1	Подробное описание . . . . .	157
6.54.2	Функции . . . . .	157
6.54.2.1	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	157
6.54.2.2	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod) . . . . .	157
6.54.2.3	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	158
6.55	Прерывания . . . . .	159
6.55.1	Подробное описание . . . . .	159
6.55.2	Функции . . . . .	159
6.55.2.1	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	159
6.55.2.2	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol) . . . . .	159
6.55.2.3	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	160
6.56	Константы . . . . .	161
6.56.1	Подробное описание . . . . .	161
6.56.2	Макросы . . . . .	161
6.56.2.1	RCC_CLK_CHANGE_TIMEOUT . . . . .	161
6.56.2.2	RCC_CLK_PLL_STABLE_TIMEOUT . . . . .	161
6.57	Типы . . . . .	162
6.57.1	Подробное описание . . . . .	163
6.57.2	Макросы . . . . .	164
6.57.2.1	IS_RCC_ADC_CLK . . . . .	164
6.57.2.2	IS_RCC_PERIPH_CLK . . . . .	164
6.57.2.3	IS_RCC_PLL_NO . . . . .	164
6.57.2.4	IS_RCC_PLL_REF . . . . .	164
6.57.2.5	IS_RCC_SPI_CLK . . . . .	165
6.57.2.6	IS_RCC_SYS_CLK . . . . .	165
6.57.2.7	IS_RCC_UART_CLK . . . . .	165
6.57.2.8	IS_RCC_USB_CLK . . . . .	165
6.57.2.9	IS_RCC_USB_FREQ . . . . .	166
6.57.3	Перечисления . . . . .	166
6.57.3.1	RCC_ADCClk_TypeDef . . . . .	166
6.57.3.2	RCC_PeriphClk_TypeDef . . . . .	166
6.57.3.3	RCC_PeriphRst_TypeDef . . . . .	167
6.57.3.4	RCC_PLLNO_TypeDef . . . . .	168
6.57.3.5	RCC_PLLRef_TypeDef . . . . .	168
6.57.3.6	RCC_SPIClk_TypeDef . . . . .	168
6.57.3.7	RCC_SysClk_TypeDef . . . . .	169

6.57.3.8	RCC_UARTClk_TypeDef	169
6.57.3.9	RCC_USBClk_TypeDef	169
6.57.3.10	RCC_USBFreq_TypeDef	169
6.58	Функции	170
6.58.1	Подробное описание	170
6.58.2	Функции	170
6.58.2.1	RCC_SysClkDiv2Out(FunctionalState State)	170
6.59	Конфигурация PLL	171
6.59.1	Подробное описание	171
6.59.2	Функции	171
6.59.2.1	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	171
6.59.2.2	RCC_PLLDeInit()	171
6.59.2.3	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	172
6.59.2.4	RCC_PLLPowerDownCmd(FunctionalState State)	173
6.59.2.5	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	173
6.60	Управление тактированием	174
6.60.1	Подробное описание	174
6.60.2	Функции	174
6.60.2.1	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	174
6.60.2.2	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	174
6.60.2.3	RCC_SysClkStatus()	175
6.61	Тактирование USB	176
6.61.1	Подробное описание	176
6.61.2	Функции	176
6.61.2.1	RCC_USBClkCmd(FunctionalState State)	176
6.61.2.2	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	176
6.62	Тактирование UART	177
6.62.1	Подробное описание	177
6.62.2	Функции	177
6.62.2.1	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	177
6.62.2.2	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	177
6.62.2.3	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	178
6.63	Тактирование SPI	180
6.63.1	Подробное описание	180
6.63.2	Функции	180
6.63.2.1	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	180

6.63.2.2	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	180
6.63.2.3	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	180
6.64	Тактирование ADC	182
6.64.1	Подробное описание	182
6.64.2	Функции	182
6.64.2.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	182
6.64.2.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	182
6.65	Управление сбросом	183
6.65.1	Подробное описание	183
6.65.2	Функции	183
6.65.2.1	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	183
6.66	Типы	184
6.66.1	Подробное описание	185
6.66.2	Макросы	185
6.66.2.1	IS_RTC_FORMAT	185
6.66.2.2	IS_RTC_MONTH	185
6.66.2.3	IS_RTC_WEEKDAY	185
6.66.3	Перечисления	185
6.66.3.1	RTC_Format_TypeDef	185
6.66.3.2	RTC_Month_TypeDef	186
6.66.3.3	RTC_Weekday_TypeDef	186
6.67	Функции	187
6.67.1	Подробное описание	187
6.68	Типы	188
6.68.1	Подробное описание	188
6.68.2	Макросы	188
6.68.2.1	IS_TIMER_EXT_INPUT	188
6.68.3	Перечисления	188
6.68.3.1	TIMER_ExtInput_TypeDef	188
6.69	Константы	189
6.70	Функции	190
6.70.1	Подробное описание	190
6.71	Конфигурация	191
6.71.1	Подробное описание	191
6.71.2	Функции	191
6.71.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	191

6.71.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	191
6.71.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	192
6.71.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	192
6.71.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	192
6.71.2.6	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	193
6.71.2.7	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	193
6.71.2.8	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	193
6.72	Прерывания	195
6.72.1	Подробное описание	195
6.72.2	Функции	195
6.72.2.1	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	195
6.72.2.2	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	195
6.72.2.3	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	195
6.73	Типы	197
6.73.1	Подробное описание	198
6.73.2	Макросы	198
6.73.2.1	IS_UART_DATA_WIDTH	198
6.73.2.2	IS_UART_DIR	198
6.73.2.3	IS_UART_FIFO_LEVEL	198
6.73.2.4	IS_UART_PARITY_BIT	198
6.73.2.5	IS_UART_STOP_BIT	199
6.73.3	Перечисления	199
6.73.3.1	UART_DataWidth_TypeDef	199
6.73.3.2	UART_Dir_Typedef	199
6.73.3.3	UART_FIFOLevel_TypeDef	199
6.73.3.4	UART_ParityBit_TypeDef	200
6.73.3.5	UART_StopBit_TypeDef	200
6.74	Константы	201
6.74.1	Подробное описание	201
6.75	Источники прерываний UART	202
6.75.1	Подробное описание	202
6.75.2	Макросы	202
6.75.2.1	UART_ITSource_All	202
6.75.2.2	UART_ITSource_ChangeCTS	202
6.75.2.3	UART_ITSource_ChangeDCD	202
6.75.2.4	UART_ITSource_ChangeDSR	202

6.75.2.5	UART_ITSource_ChangeRI . . . . .	202
6.75.2.6	UART_ITSource_ErrorBreak . . . . .	203
6.75.2.7	UART_ITSource_ErrorFrame . . . . .	203
6.75.2.8	UART_ITSource_ErrorOverflow . . . . .	203
6.75.2.9	UART_ITSource_ErrorParity . . . . .	203
6.75.2.10	UART_ITSource_RecieveTimeout . . . . .	203
6.75.2.11	UART_ITSource_RxFIFOLevel . . . . .	203
6.75.2.12	UART_ITSource_TxFIFOLevel . . . . .	203
6.76	Флаги работы UART . . . . .	204
6.76.1	Подробное описание . . . . .	204
6.76.2	Макросы . . . . .	204
6.76.2.1	UART_Flag_All . . . . .	204
6.76.2.2	UART_Flag_Busy . . . . .	204
6.76.2.3	UART_Flag_InvCTS . . . . .	204
6.76.2.4	UART_Flag_InvDCD . . . . .	204
6.76.2.5	UART_Flag_InvDSR . . . . .	204
6.76.2.6	UART_Flag_InvRI . . . . .	204
6.76.2.7	UART_Flag_RxFIFOEmpty . . . . .	205
6.76.2.8	UART_Flag_RxFIFOFull . . . . .	205
6.76.2.9	UART_Flag_TxFIFOEmpty . . . . .	205
6.76.2.10	UART_Flag_TxFIFOFull . . . . .	205
6.77	Ошибки приемника UART . . . . .	206
6.77.1	Подробное описание . . . . .	206
6.77.2	Макросы . . . . .	206
6.77.2.1	UART_Error_All . . . . .	206
6.77.2.2	UART_Error_Break . . . . .	206
6.77.2.3	UART_Error_Frame . . . . .	206
6.77.2.4	UART_Error_Overflow . . . . .	206
6.77.2.5	UART_Error_Parity . . . . .	206
6.78	Функции . . . . .	207
6.78.1	Подробное описание . . . . .	207
6.78.2	Функции . . . . .	207
6.78.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv) . . . . .	207
6.78.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State) . . . . .	207
6.78.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State) . . . . .	208
6.79	Инициализация и деинициализация . . . . .	209
6.79.1	Подробное описание . . . . .	209
6.79.2	Функции . . . . .	209
6.79.2.1	UART_DeInit(NT_UART_TypeDef *UARTx) . . . . .	209

6.79.2.2	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	209
6.79.2.3	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	210
6.80	Прием и передача	212
6.80.1	Подробное описание	212
6.80.2	Функции	212
6.80.2.1	UART_ErrorStatus(NT_UART_TypeDef *UARTx, uint32_t UART_Error)	212
6.80.2.2	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx, uint32_t UART_Error)	212
6.80.2.3	UART_FlagStatus(NT_UART_TypeDef *UARTx, uint32_t UART_Flag)	213
6.80.2.4	UART_RecieveData(NT_UART_TypeDef *UARTx)	214
6.80.2.5	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	214
6.81	Режим модема	215
6.81.1	Подробное описание	215
6.81.2	Функции	215
6.81.2.1	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	215
6.81.2.2	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	215
6.82	Прерывания	216
6.82.1	Подробное описание	216
6.82.2	Функции	216
6.82.2.1	UART_ITCmd(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource, FunctionalState State)	216
6.82.2.2	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	216
6.82.2.3	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource)	217
6.82.2.4	UART_ITRawStatus(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource)	217
6.82.2.5	UART_ITStatusClear(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource)	217
6.83	Настройка DMA	219
6.83.1	Подробное описание	219
6.83.2	Функции	219
6.83.2.1	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	219
6.83.2.2	UART_DMAMCmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	219
6.84	Константы	220
6.84.1	Подробное описание	220

6.85	Основная область флеш	221
6.85.1	Подробное описание	221
6.85.2	Макросы	221
6.85.2.1	USERFLASH_PAGE_SIZE_BYTES	221
6.85.2.2	USERFLASH_PAGE_TOTAL	221
6.85.2.3	USERFLASH_TOTAL_BYTES	221
6.86	Информационная область флеш	222
6.86.1	Подробное описание	222
6.86.2	Макросы	222
6.86.2.1	USERFLASH_INFO_PAGE_SIZE_BYTES	222
6.86.2.2	USERFLASH_INFO_PAGE_TOTAL	222
6.86.2.3	USERFLASH_INFO_TOTAL_BYTES	222
6.87	Типы	223
6.87.1	Подробное описание	223
6.87.2	Макросы	223
6.87.2.1	IS_USERFLASH_STATUS	223
6.87.3	Перечисления	223
6.87.3.1	USERFLASH_Status_TypeDef	223
6.88	Функции	224
6.88.1	Подробное описание	224
6.88.2	Функции	224
6.88.2.1	USERFLASH_Init(uint32_t SysClkFreq)	224
6.88.2.2	USERFLASH_ITCmd(FunctionalState State)	224
6.88.2.3	USERFLASH_OperationStatus()	224
6.88.2.4	USERFLASH_OperationStatusClear()	225
6.89	Основная область флеш	226
6.89.1	Подробное описание	226
6.89.2	Функции	226
6.89.2.1	USERFLASH_FullErase()	226
6.89.2.2	USERFLASH_PageErase(uint32_t PageNum)	226
6.89.2.3	USERFLASH_Read(uint32_t Address)	226
6.89.2.4	USERFLASH_Write(uint32_t Address, uint32_t Data)	227
6.90	Информационная область флеш	228
6.90.1	Подробное описание	228
6.90.2	Функции	228
6.90.2.1	USERFLASH_Info_PageErase(uint32_t PageNum)	228
6.90.2.2	USERFLASH_Info_Read(uint32_t Address)	228
6.90.2.3	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	228
6.91	Типы	230
6.91.1	Подробное описание	230



6.92 Константы . . . . .	231
6.93 Функции . . . . .	232
6.93.1 Подробное описание . . . . .	232
6.94 Конфигурация . . . . .	233
6.94.1 Подробное описание . . . . .	233
6.94.2 Функции . . . . .	233
6.94.2.1 WATCHDOG_Cmd(FunctionalState State) . . . . .	233
6.94.2.2 WATCHDOG_GetCounter() . . . . .	233
6.94.2.3 WATCHDOG_GetReload() . . . . .	233
6.94.2.4 WATCHDOG_LockCmd(FunctionalState State) . . . . .	234
6.94.2.5 WATCHDOG_RstCmd(FunctionalState State) . . . . .	234
6.94.2.6 WATCHDOG_SetReload(uint32_t ReloadVal) . . . . .	234
6.95 Прерывания . . . . .	235
6.95.1 Подробное описание . . . . .	235
6.95.2 Функции . . . . .	235
6.95.2.1 WATCHDOG_ITMaskedStatus() . . . . .	235
6.95.2.2 WATCHDOG_ITRawStatus() . . . . .	235
6.95.2.3 WATCHDOG_ITStatusClear() . . . . .	235
6.96 ADC . . . . .	236
6.96.1 Подробное описание . . . . .	236
6.97 Приватные данные . . . . .	237
6.97.1 Подробное описание . . . . .	237
6.98 Приватные константы . . . . .	238
6.98.1 Подробное описание . . . . .	238
6.99 Начальные значения регистров . . . . .	239
6.100 Приватные функции . . . . .	240
6.100.1 Подробное описание . . . . .	242
6.100.2 Функции . . . . .	242
6.100.2.1 ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State) . . . . .	242
6.100.2.2 ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State) . . . . .	242
6.100.2.3 ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	242
6.100.2.4 ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	242
6.100.2.5 ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Init_TypeDef *ADC_DC_InitStruct) . . . . .	243
6.100.2.6 ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State) . . . . .	243
6.100.2.7 ADC_DC_ITConfig(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef ADC_DC_Condition) . . . . .	243

6.100.2.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State) . . . . .	244
6.100.2.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State) . . . . .	244
6.100.2.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC_↔ _Module) . . . . .	244
6.100.2.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↔ Module) . . . . .	245
6.100.2.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ Module) . . . . .	245
6.100.2.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct) . . . . .	245
6.100.2.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	246
6.100.2.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ _Module) . . . . .	246
6.100.2.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module) . . . . .	246
6.100.2.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↔ Def *ADC_InitStruct) . . . . .	247
6.100.2.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State) . . . . .	247
6.100.2.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module) . . . . .	247
6.100.2.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State) . . . . .	248
6.100.2.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ _Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel) . . . . .	248
6.100.2.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↔ SEQ_Module) . . . . .	248
6.100.2.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef A↔ DC_SEQ_Module) . . . . .	249
6.100.2.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC_↔ _SEQ_Module) . . . . .	249
6.100.2.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module) . . . . .	249
6.100.2.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_S↔ EQ_Module) . . . . .	250
6.100.2.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef AD↔ C_SEQ_Module) . . . . .	250
6.100.2.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef AD↔ C_SEQ_Module) . . . . .	250
6.100.2.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SE↔ Q_Module) . . . . .	251
6.100.2.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SE↔ Q_Module) . . . . .	251
6.100.2.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ _Module) . . . . .	251
6.100.2.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct) . . . . .	252

6.100.2.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ Module, FunctionalState State) . . . . .	253
6.100.2.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_↵ ITCountSEQRst) . . . . .	253
6.100.2.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module) . . . . .	254
6.100.2.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_S↵ EQ_Module) . . . . .	254
6.100.2.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module) . . . . .	254
6.100.2.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SE↵ Q_Module) . . . . .	254
6.100.2.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_Init↵ Struct) . . . . .	255
6.100.2.40	ADC_SEQ_SWReq() . . . . .	255
6.100.2.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct) . . . . .	255
6.101	BOOTFLASH . . . . .	256
6.101.1	Подробное описание . . . . .	256
6.102	Приватные данные . . . . .	257
6.102.1	Подробное описание . . . . .	257
6.103	Приватные константы . . . . .	258
6.104	Приватные функции . . . . .	259
6.104.1	Подробное описание . . . . .	259
6.104.2	Функции . . . . .	259
6.104.2.1	BOOTFLASH_FullErase() . . . . .	259
6.104.2.2	BOOTFLASH_Info_PageErase(uint32_t PageNum) . . . . .	259
6.104.2.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_↵ _t Data1, uint32_t Data2, uint32_t Data3) . . . . .	260
6.104.2.4	BOOTFLASH_Init(uint32_t SysClkFreq) . . . . .	260
6.104.2.5	BOOTFLASH_ITCmd(FunctionalState State) . . . . .	260
6.104.2.6	BOOTFLASH_OperationStatus() . . . . .	260
6.104.2.7	BOOTFLASH_OperationStatusClear() . . . . .	261
6.104.2.8	BOOTFLASH_PageErase(uint32_t PageNum) . . . . .	261
6.104.2.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_↵ _t Data1, uint32_t Data2, uint32_t Data3) . . . . .	261
6.105	CAP . . . . .	262
6.105.1	Подробное описание . . . . .	262
6.106	Приватные данные . . . . .	263
6.106.1	Подробное описание . . . . .	263
6.107	Приватные константы . . . . .	264
6.108	Приватные функции . . . . .	265
6.108.1	Подробное описание . . . . .	266

6.108.2	Функции	266
6.108.2.1	CAP_Capture_Cmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	266
6.108.2.2	CAP_Capture_GetCap0(NT_CAP_TypeDef *CAPx)	266
6.108.2.3	CAP_Capture_GetCap1(NT_CAP_TypeDef *CAPx)	267
6.108.2.4	CAP_Capture_GetCap2(NT_CAP_TypeDef *CAPx)	267
6.108.2.5	CAP_Capture_GetCap3(NT_CAP_TypeDef *CAPx)	267
6.108.2.6	CAP_Capture_Init(NT_CAP_TypeDef *CAPx, CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	268
6.108.2.7	CAP_Capture_SetCap0(NT_CAP_TypeDef *CAPx, uint32_t Value)	269
6.108.2.8	CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value)	269
6.108.2.9	CAP_Capture_SetCap2(NT_CAP_TypeDef *CAPx, uint32_t Value)	269
6.108.2.10	CAP_Capture_SetCap3(NT_CAP_TypeDef *CAPx, uint32_t Value)	270
6.108.2.11	CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	270
6.108.2.12	CAP_DeInit(NT_CAP_TypeDef *CAPx)	270
6.108.2.13	CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx)	271
6.108.2.14	CAP_GetTimer(NT_CAP_TypeDef *CAPx)	271
6.108.2.15	CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_InitStruct)	271
6.108.2.16	CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)	272
6.108.2.17	CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	272
6.108.2.18	CAP_ITPendClear(NT_CAP_TypeDef *CAPx)	272
6.108.2.19	CAP_ITPendStatus(NT_CAP_TypeDef *CAPx)	272
6.108.2.20	CAP_ITStatus(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	273
6.108.2.21	CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource)	273
6.108.2.22	CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx)	273
6.108.2.23	CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx)	274
6.108.2.24	CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx)	274
6.108.2.25	CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx)	274
6.108.2.26	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct)	274
6.108.2.27	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_t CompareVal)	275
6.108.2.28	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)	275
6.108.2.29	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_t CompareVal)	275
6.108.2.30	CAP_PWM_SetShadowPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal)	276
6.108.2.31	CAP_PWM_StructInit(CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct)	276

6.108.2.32	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	276
6.108.2.33	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal)	276
6.108.2.34	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct)	277
6.108.2.35	CAP_SwSync(NT_CAP_TypeDef *CAPx)	277
6.108.2.36	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	277
6.108.2.37	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	277
6.109	DMA	279
6.109.1	Подробное описание	279
6.110	Приватные данные	280
6.110.1	Подробное описание	280
6.111	Приватные константы	281
6.111.1	Подробное описание	281
6.112	Начальные значения регистров	282
6.113	Приватные функции	283
6.113.1	Подробное описание	283
6.113.2	Функции	283
6.113.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	283
6.113.2.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	284
6.113.2.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	284
6.113.2.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	284
6.113.2.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	285
6.113.2.6	DMA_ClearErrorStatus()	285
6.113.2.7	DMA_DeInit()	286
6.113.2.8	DMA_ErrorStatus()	287
6.113.2.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	287
6.113.2.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	287
6.113.2.11	DMA_MasterEnableCmd(FunctionalState State)	288
6.113.2.12	DMA_MasterEnableStatus()	288
6.113.2.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	288
6.113.2.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	288
6.113.2.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	289
6.113.2.16	DMA_StateStatus()	289
6.113.2.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	289
6.113.2.18	DMA_SWRequestCmd(uint32_t DMA_Channel)	290
6.113.2.19	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	290
6.113.2.20	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	290
6.114	EXTMEM	291
6.114.1	Подробное описание	291

6.115	Приватные данные	292
6.115.1	Подробное описание	292
6.116	Приватные константы	293
6.116.1	Подробное описание	293
6.117	Начальные значения регистров	294
6.117.1	Подробное описание	294
6.117.2	Макросы	294
6.117.2.1	EXT_MEM_CFG_Reset_Value	294
6.118	Приватные функции	295
6.118.1	Подробное описание	295
6.118.2	Функции	295
6.118.2.1	EXTMEM_DeInit()	295
6.118.2.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	295
6.118.2.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	295
6.119	GPIO	297
6.119.1	Подробное описание	297
6.120	Приватные данные	298
6.120.1	Подробное описание	298
6.121	Приватные константы	299
6.121.1	Подробное описание	299
6.122	Начальные значения регистров	300
6.122.1	Подробное описание	300
6.122.2	Макросы	300
6.122.2.1	GPIO_DATAOUT_Reset_Value	300
6.122.2.2	GPIO_GPIODEN0_Reset_Value	300
6.122.2.3	GPIO_GPIODEN1_Reset_Value	300
6.122.2.4	GPIO_GPIODEN2_Reset_Value	300
6.122.2.5	GPIO_GPIODEN3_Reset_Value	301
6.122.2.6	GPIO_GPIOODCTLx_Reset_Value	301
6.122.2.7	GPIO_GPIOODSCTLx_Reset_Value	301
6.122.2.8	GPIO_GPIOPCTLx_Reset_Value	301
6.122.2.9	GPIO_GPIOPUCTLx_Reset_Value	301
6.122.2.10	GPIO_GPIOQEx_Reset_Value	301
6.122.2.11	GPIO_GPIOQMx_Reset_Value	301
6.122.2.12	GPIO_GPIOQPx_Reset_Value	301
6.122.2.13	GPIO_GPIOSEx_Reset_Value	302
6.123	Маски портов	303
6.123.1	Подробное описание	303
6.123.2	Макросы	303
6.123.2.1	GPIO_Regs_A_C_E_G_Mask	303

6.123.2.2 GPIO_Regs_B_D_F_H_Mask . . . . .	303
6.123.2.3 GPIO_Regs_GPIOA_Mask . . . . .	303
6.123.2.4 GPIO_Regs_GPIOB_Mask . . . . .	303
6.123.2.5 GPIO_Regs_GPIOC_Mask . . . . .	303
6.123.2.6 GPIO_Regs_GPIOD_Mask . . . . .	303
6.123.2.7 GPIO_Regs_GPIOE_Mask . . . . .	304
6.123.2.8 GPIO_Regs_GPIOF_Mask . . . . .	304
6.123.2.9 GPIO_Regs_GPIOG_Mask . . . . .	304
6.123.2.10 GPIO_Regs_GPIOH_Mask . . . . .	304
6.124 Приватные функции . . . . .	305
6.124.1 Подробное описание . . . . .	306
6.124.2 Функции . . . . .	306
6.124.2.1 GPIO_AltFuncConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc) . . . . .	306
6.124.2.2 GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	307
6.124.2.3 GPIO_DeInit(NT_GPIO_TypeDef *GPIOx) . . . . .	307
6.124.2.4 GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct) . . . . .	307
6.124.2.5 GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	308
6.124.2.6 GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol) . . . . .	308
6.124.2.7 GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	309
6.124.2.8 GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	309
6.124.2.9 GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod) . . . . .	309
6.124.2.10 GPIO_Read(NT_GPIO_TypeDef *GPIOx) . . . . .	310
6.124.2.11 GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	310
6.124.2.12 GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal) . . . . .	310
6.124.2.13 GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	311
6.124.2.14 GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct) . . . . .	311
6.124.2.15 GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	311
6.124.2.16 GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	312
6.124.2.17 GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal) . . . . .	312
6.124.2.18 GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal) . . . . .	312
6.124.2.19 GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal) . . . . .	313
6.125 RCC . . . . .	314
6.125.1 Подробное описание . . . . .	314

6.126	Приватные данные	315
6.126.1	Подробное описание	315
6.127	Приватные константы	316
6.127.1	Подробное описание	316
6.128	Начальные значения регистров	317
6.128.1	Подробное описание	317
6.128.2	Макросы	317
6.128.2.1	RCC_PLL_CTRL_Reset_Value	317
6.128.2.2	RCC_PLL_NF_Reset_Value	317
6.128.2.3	RCC_PLL_NR_Reset_Value	317
6.128.2.4	RCC_PLL_OD_Reset_Value	317
6.129	Приватные функции	318
6.129.1	Подробное описание	319
6.129.2	Функции	319
6.129.2.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	319
6.129.2.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	319
6.129.2.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	319
6.129.2.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	320
6.129.2.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	320
6.129.2.6	RCC_PLLDeInit()	321
6.129.2.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	321
6.129.2.8	RCC_PLLPowerDownCmd(FunctionalState State)	322
6.129.2.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	322
6.129.2.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	322
6.129.2.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	323
6.129.2.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	323
6.129.2.13	RCC_SysClkDiv2Out(FunctionalState State)	323
6.129.2.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	324
6.129.2.15	RCC_SysClkStatus()	324
6.129.2.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	324
6.129.2.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	325
6.129.2.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	325
6.129.2.19	RCC_USBClkCmd(FunctionalState State)	325



6.129.2.2	RCC_USBCLKConfig(RCC_USBCLK_TypeDef RCC_USBCLK, RCC_↵ USBFreq_TypeDef RCC_USBFreq) . . . . .	326
6.129.2.2	RCC_WaitClkChange(RCC_SysClk_TypeDef RCC_SysClk) . . . . .	326
6.130	RTC . . . . .	327
6.130.1	Подробное описание . . . . .	327
6.131	Приватные данные . . . . .	328
6.131.1	Подробное описание . . . . .	328
6.132	Приватные функции . . . . .	329
6.132.1	Подробное описание . . . . .	329
6.133	TIMER . . . . .	330
6.133.1	Подробное описание . . . . .	330
6.134	Приватные данные . . . . .	331
6.134.1	Подробное описание . . . . .	331
6.135	Приватные константы . . . . .	332
6.136	Приватные функции . . . . .	333
6.136.1	Подробное описание . . . . .	333
6.136.2	Функции . . . . .	333
6.136.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State) . . . . .	333
6.136.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_↵ _ExtInput_TypeDef TIMER_ExtInput) . . . . .	333
6.136.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_↵ t TimerClkFreq, uint32_t TimerFreq) . . . . .	334
6.136.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx) . . . . .	334
6.136.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx) . . . . .	334
6.136.2.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State) . . . . .	335
6.136.2.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx) . . . . .	335
6.136.2.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx) . . . . .	335
6.136.2.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_↵ t TimerClkFreq, uint32_t TimerPeriod) . . . . .	336
6.136.2.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_↵ t CounterVal) . . . . .	336
6.136.2.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_↵ t ReloadVal) . . . . .	336
6.137	UART . . . . .	337
6.137.1	Подробное описание . . . . .	337
6.138	Приватные данные . . . . .	338
6.138.1	Подробное описание . . . . .	338
6.139	Приватные константы . . . . .	339
6.140	Приватные функции . . . . .	340
6.140.1	Подробное описание . . . . .	341
6.140.2	Функции . . . . .	341

6.140.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	341
6.140.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	341
6.140.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	341
6.140.2.4	UART_DeInit(NT_UART_TypeDef *UARTx)	342
6.140.2.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	343
6.140.2.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	343
6.140.2.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, uint32_t UART_Error)	343
6.140.2.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx, uint32_t UART_Error)	344
6.140.2.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, uint32_t UART_Flag)	344
6.140.2.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	344
6.140.2.11	UART_ITCmd(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource, FunctionalState State)	345
6.140.2.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	345
6.140.2.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource)	346
6.140.2.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource)	347
6.140.2.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, uint32_t UART_ITSource)	347
6.140.2.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	347
6.140.2.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	348
6.140.2.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	348
6.140.2.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	348
6.140.2.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	349
6.141	USERFLASH	350
6.141.1	Подробное описание	350
6.142	Приватные данные	351
6.142.1	Подробное описание	351
6.143	Приватные константы	352
6.144	Приватные функции	353
6.144.1	Подробное описание	353
6.144.2	Функции	353
6.144.2.1	USERFLASH_FullErase()	353

6.144.2.2 USERFLASH_Info_PageErase(uint32_t PageNum) . . . . .	353
6.144.2.3 USERFLASH_Info_Read(uint32_t Address) . . . . .	354
6.144.2.4 USERFLASH_Info_Write(uint32_t Address, uint32_t Data) . . . . .	354
6.144.2.5 USERFLASH_Init(uint32_t SysClkFreq) . . . . .	354
6.144.2.6 USERFLASH_ITCmd(FunctionalState State) . . . . .	354
6.144.2.7 USERFLASH_OperationStatus() . . . . .	355
6.144.2.8 USERFLASH_OperationStatusClear() . . . . .	355
6.144.2.9 USERFLASH_PageErase(uint32_t PageNum) . . . . .	355
6.144.2.10 USERFLASH_Read(uint32_t Address) . . . . .	355
6.144.2.11 USERFLASH_Write(uint32_t Address, uint32_t Data) . . . . .	356
6.145 WATCHDOG . . . . .	357
6.145.1 Подробное описание . . . . .	357
6.146 Приватные данные . . . . .	358
6.146.1 Подробное описание . . . . .	358
6.147 Приватные константы . . . . .	359
6.147.1 Подробное описание . . . . .	359
6.147.2 Макросы . . . . .	359
6.147.2.1 WATCHDOG_Lock_Value . . . . .	359
6.147.2.2 WATCHDOG_Unlock_Value . . . . .	359
6.148 Приватные функции . . . . .	360
6.148.1 Подробное описание . . . . .	360
6.148.2 Функции . . . . .	360
6.148.2.1 WATCHDOG_Cmd(FunctionalState State) . . . . .	360
6.148.2.2 WATCHDOG_GetCounter() . . . . .	360
6.148.2.3 WATCHDOG_GetReload() . . . . .	360
6.148.2.4 WATCHDOG_ITMaskedStatus() . . . . .	361
6.148.2.5 WATCHDOG_ITRawStatus() . . . . .	361
6.148.2.6 WATCHDOG_ITStatusClear() . . . . .	361
6.148.2.7 WATCHDOG_LockCmd(FunctionalState State) . . . . .	361
6.148.2.8 WATCHDOG_RstCmd(FunctionalState State) . . . . .	361
6.148.2.9 WATCHDOG_SetReload(uint32_t ReloadVal) . . . . .	362
6.149 Драйвер периферии . . . . .	363
6.149.1 Подробное описание . . . . .	363
6.150 Настройка драйвера . . . . .	364
6.150.1 Подробное описание . . . . .	364
6.150.2 Макросы . . . . .	364
6.150.2.1 EXT_OSC_VALUE . . . . .	364
6.150.2.2 INT_OSC_VALUE . . . . .	364
6.151 Макросы . . . . .	365
6.151.1 Подробное описание . . . . .	365

6.152	Типы	366
6.152.1	Подробное описание	366
6.152.2	Макросы	366
6.152.2.1	IS_CAP_ALL_PERIPH	366
6.152.2.2	IS_GPIO_ALL_PERIPH	367
6.152.2.3	IS_SPI_ALL_PERIPH	367
6.152.2.4	IS_TIMER_ALL_PERIPH	367
6.152.2.5	IS_UART_ALL_PERIPH	367
6.153	Периферия	369
6.153.1	Подробное описание	369
7	Структуры данных	371
7.1	Структура _CHANNEL_CFG_bits	371
7.1.1	Подробное описание	371
7.1.2	Поля	371
7.1.2.1	CYCLE_CTRL	371
7.1.2.2	DST_INC	372
7.1.2.3	DST_PROT_BUFFERABLE	372
7.1.2.4	DST_PROT_CACHEABLE	372
7.1.2.5	DST_PROT_PRIVILEGED	372
7.1.2.6	DST_SIZE	372
7.1.2.7	N_MINUS_1	372
7.1.2.8	NEXT_USEBURST	372
7.1.2.9	R_POWER	372
7.1.2.10	SRC_INC	373
7.1.2.11	SRC_PROT_BUFFERABLE	373
7.1.2.12	SRC_PROT_CACHEABLE	373
7.1.2.13	SRC_PROT_PRIVILEGED	373
7.1.2.14	SRC_SIZE	373
7.2	Структура ADC_DC_Init_TypeDef	373
7.2.1	Подробное описание	374
7.2.2	Поля	374
7.2.2.1	ADC_DC_Channel	374
7.2.2.2	ADC_DC_Condition	374
7.2.2.3	ADC_DC_Mode	374
7.2.2.4	ADC_DC_ThresholdHigh	374
7.2.2.5	ADC_DC_ThresholdLow	374
7.3	Структура ADC_Init_TypeDef	375
7.3.1	Подробное описание	375
7.3.2	Поля	375

7.3.2.1	ADC_Average	375
7.3.2.2	ADC_Measure_A	375
7.3.2.3	ADC_Measure_B	375
7.3.2.4	ADC_Mode	376
7.3.2.5	ADC_Phase	376
7.3.2.6	ADC_Resolution	376
7.4	Структура ADC_SEQ_Init_TypeDef	376
7.4.1	Подробное описание	376
7.4.2	Поля	376
7.4.2.1	ADC_Channels	376
7.4.2.2	ADC_SEQ_ConversionCount	377
7.4.2.3	ADC_SEQ_ConversionDelay	377
7.4.2.4	ADC_SEQ_DC	377
7.4.2.5	ADC_SEQ_StartEvent	377
7.4.2.6	ADC_SEQ_SWReqEn	377
7.5	Структура CAP_Capture_Init_TypeDef	377
7.5.1	Подробное описание	378
7.5.2	Поля	378
7.5.2.1	CAP_Capture_PolarityEvent0	378
7.5.2.2	CAP_Capture_PolarityEvent1	378
7.5.2.3	CAP_Capture_PolarityEvent2	378
7.5.2.4	CAP_Capture_PolarityEvent3	378
7.5.2.5	CAP_Capture_Prescale	379
7.5.2.6	CAP_Capture_RstEvent0	379
7.5.2.7	CAP_Capture_RstEvent1	379
7.5.2.8	CAP_Capture_RstEvent2	379
7.5.2.9	CAP_Capture_RstEvent3	379
7.5.2.10	CAP_Capture_StopVal	379
7.5.2.11	CAP_CaptureMode	379
7.6	Структура CAP_Init_TypeDef	380
7.6.1	Подробное описание	380
7.6.2	Поля	380
7.6.2.1	CAP_Halt	380
7.6.2.2	CAP_Mode	380
7.6.2.3	CAP_SyncCmd	380
7.6.2.4	CAP_SyncOut	381
7.7	Структура CAP_PWM_Init_TypeDef	381
7.7.1	Подробное описание	381
7.7.2	Поля	381
7.7.2.1	CAP_PWM_Compare	381

7.7.2.2	CAP_PWM_Period	381
7.7.2.3	CAP_PWM_Polarity	381
7.8	Структура DMA_Channel_TypeDef	382
7.8.1	Подробное описание	382
7.8.2	Поля	382
7.8.2.1	CHANNEL_CFG	382
7.8.2.2	CHANNEL_CFG_bit	382
7.8.2.3	DST_DATA_END	382
7.8.2.4	SRC_DATA_END	382
7.9	Структура DMA_ChannelInit_TypeDef	383
7.9.1	Подробное описание	383
7.9.2	Поля	383
7.9.2.1	DMA_ArbitrationRate	383
7.9.2.2	DMA_DstDataEndPtr	383
7.9.2.3	DMA_DstDataInc	384
7.9.2.4	DMA_DstDataSize	384
7.9.2.5	DMA_DstProtect	384
7.9.2.6	DMA_Mode	384
7.9.2.7	DMA_NextUseburst	384
7.9.2.8	DMA_SrcDataEndPtr	384
7.9.2.9	DMA_SrcDataInc	384
7.9.2.10	DMA_SrcDataSize	385
7.9.2.11	DMA_SrcProtect	385
7.9.2.12	DMA_TransfersTotal	385
7.10	Структура DMA_ConfigData_TypeDef	385
7.10.1	Подробное описание	385
7.10.2	Поля	386
7.10.2.1	ALT_DATA	386
7.10.2.2	PRM_DATA	386
7.10.2.3	RESERVED0	386
7.10.2.4	RESERVED1	386
7.11	Структура DMA_ConfigStruct_TypeDef	386
7.11.1	Подробное описание	386
7.11.2	Поля	386
7.11.2.1	CH	386
7.12	Структура DMA_Init_TypeDef	387
7.12.1	Подробное описание	387
7.12.2	Поля	387
7.12.2.1	DMA_Channel	387
7.12.2.2	DMA_ChannelEnable	387

7.12.2.3	DMA_HighPriority	387
7.12.2.4	DMA_PrmAlt	387
7.12.2.5	DMA_Protection	388
7.12.2.6	DMA_ReqMask	388
7.12.2.7	DMA_UseBurst	388
7.13	Структура DMA_Protect_TypeDef	388
7.13.1	Подробное описание	388
7.13.2	Поля	388
7.13.2.1	BUFFERABLE	388
7.13.2.2	CACHEABLE	389
7.13.2.3	PRIVELGED	389
7.14	Структура EXTMEM_Init_TypeDef	389
7.14.1	Подробное описание	389
7.14.2	Поля	389
7.14.2.1	CEMask	389
7.14.2.2	EXTMEM_ReadWaitState	389
7.14.2.3	EXTMEM_RWWaitState	390
7.14.2.4	EXTMEM_Width	390
7.14.2.5	EXTMEM_WriteWaitState	390
7.15	Структура GPIO_Init_TypeDef	390
7.15.1	Подробное описание	390
7.15.2	Поля	391
7.15.2.1	GPIO_AltFunc	391
7.15.2.2	GPIO_Dir	391
7.15.2.3	GPIO_Load	391
7.15.2.4	GPIO_Mode	391
7.15.2.5	GPIO_Out	391
7.15.2.6	GPIO_OutMode	391
7.15.2.7	GPIO_Pin	391
7.15.2.8	GPIO_PullUp	392
7.16	Структура RCC_PLLInit_TypeDef	392
7.16.1	Подробное описание	392
7.16.2	Поля	392
7.16.2.1	RCC_PLLDiv	392
7.16.2.2	RCC_PLLNF	392
7.16.2.3	RCC_PLLNO	393
7.16.2.4	RCC_PLLNR	393
7.16.2.5	RCC_PLLRef	393
7.17	Структура RTC_Date_TypeDef	393
7.17.1	Подробное описание	393

7.17.2 Поля . . . . .	393
7.17.2.1 RTC_Day . . . . .	393
7.17.2.2 RTC_Month . . . . .	394
7.17.2.3 RTC_Weekday . . . . .	394
7.17.2.4 RTC_Year . . . . .	394
7.18 Структура RTC_Time_TypeDef . . . . .	394
7.18.1 Подробное описание . . . . .	394
7.18.2 Поля . . . . .	394
7.18.2.1 RTC_Hour . . . . .	394
7.18.2.2 RTC_Minute . . . . .	395
7.18.2.3 RTC_Psecond . . . . .	395
7.18.2.4 RTC_Second . . . . .	395
7.19 Структура UART_Init_TypeDef . . . . .	395
7.19.1 Подробное описание . . . . .	395
7.19.2 Поля . . . . .	395
7.19.2.1 UART_BaudRate . . . . .	395
7.19.2.2 UART_ClkFreq . . . . .	396
7.19.2.3 UART_DataWidth . . . . .	396
7.19.2.4 UART_FIFOEn . . . . .	396
7.19.2.5 UART_FIFOLevelRx . . . . .	396
7.19.2.6 UART_FIFOLevelTx . . . . .	396
7.19.2.7 UART_ParityBit . . . . .	396
7.19.2.8 UART_RxEn . . . . .	396
7.19.2.9 UART_StopBit . . . . .	397
7.19.2.10 UART_TxEn . . . . .	397
7.20 Структура UART_ModemInit_TypeDef . . . . .	397
7.20.1 Подробное описание . . . . .	397
7.20.2 Поля . . . . .	397
7.20.2.1 UART_CTSEn . . . . .	397
7.20.2.2 UART_InvDTR . . . . .	398
7.20.2.3 UART_InvRTS . . . . .	398
7.20.2.4 UART_RTSEn . . . . .	398
8 Файлы . . . . .	399
8.1 Файл niietcm4.h . . . . .	399
8.1.1 Подробное описание . . . . .	400
8.1.2 Макросы . . . . .	401
8.1.2.1 EXT_OSC_VALUE . . . . .	401
8.1.2.2 INT_OSC_VALUE . . . . .	401
8.1.2.3 IS_CAP_ALL_PERIPH . . . . .	401



8.1.2.4	IS_GPIO_ALL_PERIPH . . . . .	401
8.1.2.5	IS_SPI_ALL_PERIPH . . . . .	402
8.1.2.6	IS_TIMER_ALL_PERIPH . . . . .	402
8.1.2.7	IS_UART_ALL_PERIPH . . . . .	402
8.2	Файл niietcm4_adc.c . . . . .	403
8.2.1	Подробное описание . . . . .	405
8.2.2	Функции . . . . .	405
8.2.2.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State) . . . . .	405
8.2.2.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State) . . . . .	406
8.2.2.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	406
8.2.2.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↵ Module) . . . . .	406
8.2.2.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↵ DC_DC_Init_TypeDef *ADC_DC_InitStruct) . . . . .	407
8.2.2.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State) . . . . .	407
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) . . . . .	407
8.2.2.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↵ Module, FunctionalState State) . . . . .	408
8.2.2.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↵ Module, FunctionalState State) . . . . .	408
8.2.2.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC↵ _Module) . . . . .	408
8.2.2.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↵ Module) . . . . .	409
8.2.2.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↵ Module) . . . . .	409
8.2.2.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct) . . . . .	409
8.2.2.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	410
8.2.2.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC↵ _Module) . . . . .	410
8.2.2.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module) . . . . .	410
8.2.2.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↵ Def *ADC_InitStruct) . . . . .	411
8.2.2.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State) . . . . .	412
8.2.2.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module) . . . . .	412
8.2.2.20	ADC_SEQ_DMACmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ Module, FunctionalState State) . . . . .	412
8.2.2.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ↵ _Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel) . . . . .	413

8.2.2.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	413
8.2.2.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	413
8.2.2.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	414
8.2.2.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	414
8.2.2.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	414
8.2.2.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	415
8.2.2.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	415
8.2.2.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	415
8.2.2.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	416
8.2.2.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	417
8.2.2.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	417
8.2.2.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	417
8.2.2.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	418
8.2.2.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	418
8.2.2.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	418
8.2.2.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	419
8.2.2.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	419
8.2.2.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	419
8.2.2.40	ADC_SEQ_SWReq()	420
8.2.2.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	420
8.3	Файл niietcm4_adc.h	420
8.3.1	Подробное описание	426
8.3.2	Макросы	427
8.3.2.1	ADC_Channel_0	427
8.3.2.2	ADC_Channel_1	427
8.3.2.3	ADC_Channel_10	427
8.3.2.4	ADC_Channel_11	427

8.3.2.5	ADC_Channel_12	427
8.3.2.6	ADC_Channel_13	427
8.3.2.7	ADC_Channel_14	427
8.3.2.8	ADC_Channel_15	427
8.3.2.9	ADC_Channel_16	428
8.3.2.10	ADC_Channel_17	428
8.3.2.11	ADC_Channel_18	428
8.3.2.12	ADC_Channel_19	428
8.3.2.13	ADC_Channel_2	428
8.3.2.14	ADC_Channel_20	428
8.3.2.15	ADC_Channel_21	428
8.3.2.16	ADC_Channel_22	428
8.3.2.17	ADC_Channel_23	428
8.3.2.18	ADC_Channel_3	428
8.3.2.19	ADC_Channel_4	429
8.3.2.20	ADC_Channel_5	429
8.3.2.21	ADC_Channel_6	429
8.3.2.22	ADC_Channel_7	429
8.3.2.23	ADC_Channel_8	429
8.3.2.24	ADC_Channel_9	429
8.3.2.25	ADC_Channel_All	429
8.3.2.26	ADC_Channel_None	429
8.3.2.27	ADC_DC_0	429
8.3.2.28	ADC_DC_1	430
8.3.2.29	ADC_DC_10	430
8.3.2.30	ADC_DC_11	430
8.3.2.31	ADC_DC_12	430
8.3.2.32	ADC_DC_13	430
8.3.2.33	ADC_DC_14	430
8.3.2.34	ADC_DC_15	430
8.3.2.35	ADC_DC_16	430
8.3.2.36	ADC_DC_17	430
8.3.2.37	ADC_DC_18	430
8.3.2.38	ADC_DC_19	431
8.3.2.39	ADC_DC_2	431
8.3.2.40	ADC_DC_20	431
8.3.2.41	ADC_DC_21	431
8.3.2.42	ADC_DC_22	431
8.3.2.43	ADC_DC_23	431
8.3.2.44	ADC_DC_3	431

8.3.2.45	ADC_DC_4	431
8.3.2.46	ADC_DC_5	431
8.3.2.47	ADC_DC_6	432
8.3.2.48	ADC_DC_7	432
8.3.2.49	ADC_DC_8	432
8.3.2.50	ADC_DC_9	432
8.3.2.51	ADC_DC_All	432
8.3.2.52	ADC_DC_None	432
8.3.2.53	ADC_SEQ_0	432
8.3.2.54	ADC_SEQ_1	432
8.3.2.55	ADC_SEQ_2	432
8.3.2.56	ADC_SEQ_3	433
8.3.2.57	ADC_SEQ_4	433
8.3.2.58	ADC_SEQ_5	433
8.3.2.59	ADC_SEQ_6	433
8.3.2.60	ADC_SEQ_7	433
8.3.2.61	IS_ADC_AVERAGE	433
8.3.2.62	IS_ADC_DC_CHANNEL	433
8.3.2.63	IS_ADC_DC_CONDITION	434
8.3.2.64	IS_ADC_DC_MODE	434
8.3.2.65	IS_ADC_DC_MODULE	434
8.3.2.66	IS_ADC_MEASURE	435
8.3.2.67	IS_ADC_MODE	435
8.3.2.68	IS_ADC_MODULE	435
8.3.2.69	IS_ADC_RESOLUTION	436
8.3.2.70	IS_ADC_SEQ_FIFO_LEVEL	436
8.3.2.71	IS_ADC_SEQ_MODULE	436
8.3.2.72	IS_ADC_SEQ_START_EVENT	436
8.3.3	Перечисления	437
8.3.3.1	ADC_Average_TypeDef	437
8.3.3.2	ADC_DC_Channel_TypeDef	437
8.3.3.3	ADC_DC_Condition_TypeDef	438
8.3.3.4	ADC_DC_Mode_TypeDef	438
8.3.3.5	ADC_DC_Module_TypeDef	439
8.3.3.6	ADC_Measure_TypeDef	439
8.3.3.7	ADC_Mode_TypeDef	439
8.3.3.8	ADC_Module_TypeDef	440
8.3.3.9	ADC_Resolution_TypeDef	440
8.3.3.10	ADC_SEQ_FIFOLevel_TypeDef	440
8.3.3.11	ADC_SEQ_Module_TypeDef	441

8.3.3.12	ADC_SEQ_StartEvent_TypeDef . . . . .	441
8.3.4	Функции . . . . .	441
8.3.4.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State) . . . . .	441
8.3.4.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State) . . . . .	442
8.3.4.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	442
8.3.4.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↔ Module) . . . . .	442
8.3.4.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↔ DC_DC_Init_TypeDef *ADC_DC_InitStruct) . . . . .	443
8.3.4.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State) . . . . .	444
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) . . . . .	444
8.3.4.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State) . . . . .	445
8.3.4.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State) . . . . .	446
8.3.4.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC↔_Module) . . . . .	446
8.3.4.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↔ Module) . . . . .	446
8.3.4.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ Module) . . . . .	447
8.3.4.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct) . . . . .	447
8.3.4.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module) . . . . .	447
8.3.4.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC↔_Module) . . . . .	448
8.3.4.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module) . . . . .	448
8.3.4.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↔ Def *ADC_InitStruct) . . . . .	448
8.3.4.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State) . . . . .	449
8.3.4.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module) . . . . .	449
8.3.4.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State) . . . . .	449
8.3.4.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ↔_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel) . . . . .	449
8.3.4.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↔ SEQ_Module) . . . . .	450
8.3.4.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef A↔ DC_SEQ_Module) . . . . .	450
8.3.4.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC↔_SEQ_Module) . . . . .	450

8.3.4.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	451
8.3.4.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	451
8.3.4.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	451
8.3.4.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	452
8.3.4.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	453
8.3.4.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	453
8.3.4.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	453
8.3.4.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	454
8.3.4.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	454
8.3.4.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	454
8.3.4.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	455
8.3.4.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	455
8.3.4.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	455
8.3.4.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	456
8.3.4.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	456
8.3.4.40	ADC_SEQ_SWReq()	456
8.3.4.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	457
8.4	Файл nietscm4_bootflash.c	458
8.4.1	Подробное описание	458
8.4.2	Функции	459
8.4.2.1	BOOTFLASH_FullErase()	459
8.4.2.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	459
8.4.2.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	459
8.4.2.4	BOOTFLASH_Init(uint32_t SysClkFreq)	460
8.4.2.5	BOOTFLASH_ITCmd(FunctionalState State)	460
8.4.2.6	BOOTFLASH_OperationStatus()	460
8.4.2.7	BOOTFLASH_OperationStatusClear()	460
8.4.2.8	BOOTFLASH_PageErase(uint32_t PageNum)	461

8.4.2.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	461
8.5	Файл niietcm4_bootflash.h	461
8.5.1	Подробное описание	462
8.5.2	Макросы	463
8.5.2.1	BOOTFLASH_INFO_PAGE_SIZE_BYTES	463
8.5.2.2	BOOTFLASH_INFO_PAGE_TOTAL	463
8.5.2.3	BOOTFLASH_INFO_TOTAL_BYTES	463
8.5.2.4	BOOTFLASH_PAGE_SIZE_BYTES	463
8.5.2.5	BOOTFLASH_PAGE_TOTAL	464
8.5.2.6	BOOTFLASH_TOTAL_BYTES	464
8.5.2.7	IS_BOOTFLASH_STATUS	464
8.5.3	Перечисления	464
8.5.3.1	BOOTFLASH_Status_TypeDef	464
8.5.4	Функции	464
8.5.4.1	BOOTFLASH_FullErase()	464
8.5.4.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	464
8.5.4.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	465
8.5.4.4	BOOTFLASH_Init(uint32_t SysClkFreq)	465
8.5.4.5	BOOTFLASH_ITCmd(FunctionalState State)	465
8.5.4.6	BOOTFLASH_OperationStatus()	466
8.5.4.7	BOOTFLASH_OperationStatusClear()	466
8.5.4.8	BOOTFLASH_PageErase(uint32_t PageNum)	466
8.5.4.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	466
8.6	Файл niietcm4_cap.c	466
8.6.1	Подробное описание	468
8.6.2	Функции	469
8.6.2.1	CAP_Capture_Cmd(NT_CAP_TypeDef *CAPx, FunctionalState State)	469
8.6.2.2	CAP_Capture_GetCap0(NT_CAP_TypeDef *CAPx)	469
8.6.2.3	CAP_Capture_GetCap1(NT_CAP_TypeDef *CAPx)	469
8.6.2.4	CAP_Capture_GetCap2(NT_CAP_TypeDef *CAPx)	470
8.6.2.5	CAP_Capture_GetCap3(NT_CAP_TypeDef *CAPx)	470
8.6.2.6	CAP_Capture_Init(NT_CAP_TypeDef *CAPx, CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct)	470
8.6.2.7	CAP_Capture_SetCap0(NT_CAP_TypeDef *CAPx, uint32_t Value)	471
8.6.2.8	CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value)	471
8.6.2.9	CAP_Capture_SetCap2(NT_CAP_TypeDef *CAPx, uint32_t Value)	471
8.6.2.10	CAP_Capture_SetCap3(NT_CAP_TypeDef *CAPx, uint32_t Value)	471

8.6.2.11	CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_Capture_InitStruct) . . . . .	472
8.6.2.12	CAP_DeInit(NT_CAP_TypeDef *CAPx) . . . . .	472
8.6.2.13	CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx) . . . . .	472
8.6.2.14	CAP_GetTimer(NT_CAP_TypeDef *CAPx) . . . . .	473
8.6.2.15	CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_InitStruct) . . . . .	473
8.6.2.16	CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State) . . . . .	473
8.6.2.17	CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource) . . . . .	474
8.6.2.18	CAP_ITPendClear(NT_CAP_TypeDef *CAPx) . . . . .	474
8.6.2.19	CAP_ITPendStatus(NT_CAP_TypeDef *CAPx) . . . . .	474
8.6.2.20	CAP_ITStatus(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource) . . . . .	474
8.6.2.21	CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource) . . . . .	475
8.6.2.22	CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx) . . . . .	475
8.6.2.23	CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx) . . . . .	475
8.6.2.24	CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx) . . . . .	476
8.6.2.25	CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx) . . . . .	476
8.6.2.26	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct) . . . . .	476
8.6.2.27	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_t CompareVal) . . . . .	476
8.6.2.28	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal) . . . . .	477
8.6.2.29	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_t CompareVal) . . . . .	477
8.6.2.30	CAP_PWM_SetShadowPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal) . . . . .	477
8.6.2.31	CAP_PWM_StructInit(CAP_PWM_Init_TypeDef *CAP_PWM_InitStruct) . . . . .	478
8.6.2.32	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal) . . . . .	479
8.6.2.33	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal) . . . . .	479
8.6.2.34	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct) . . . . .	479
8.6.2.35	CAP_SwSync(NT_CAP_TypeDef *CAPx) . . . . .	480
8.6.2.36	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State) . . . . .	480
8.6.2.37	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State) . . . . .	480
8.7	Файл niectm4_cap.h . . . . .	480
8.7.1	Подробное описание . . . . .	483
8.7.2	Макросы . . . . .	484
8.7.2.1	CAP_ITSource_All . . . . .	484
8.7.2.2	CAP_ITSource_CapEvent0 . . . . .	484
8.7.2.3	CAP_ITSource_CapEvent1 . . . . .	484



8.7.2.4	CAP_ITSource_CapEvent2	484
8.7.2.5	CAP_ITSource_CapEvent3	484
8.7.2.6	CAP_ITSource_GeneralInt	484
8.7.2.7	CAP_ITSource_TimerEqCompare	485
8.7.2.8	CAP_ITSource_TimerEqPeriod	485
8.7.2.9	CAP_ITSource_TimerOvf	485
8.7.2.10	IS_CAP_CAPTURE_MODE	485
8.7.2.11	IS_CAP_CAPTURE_POLARITY	485
8.7.2.12	IS_CAP_HALT	485
8.7.2.13	IS_CAP_MODE	486
8.7.2.14	IS_CAP_PWM_POLARITY	486
8.7.2.15	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	487
8.7.3.4	CAP_Mode_TypeDef	487
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)	488
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)	489
8.7.4.8	CAP_Capture_SetCap1(NT_CAP_TypeDef *CAPx, uint32_t Value)	489
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)	490
8.7.4.11	CAP_Capture_StructInit(CAP_Capture_Init_TypeDef *CAP_↔ Capture_InitStruct)	490
8.7.4.12	CAP_DeInit(NT_CAP_TypeDef *CAPx)	491
8.7.4.13	CAP_GetShadowTimer(NT_CAP_TypeDef *CAPx)	491
8.7.4.14	CAP_GetTimer(NT_CAP_TypeDef *CAPx)	491
8.7.4.15	CAP_Init(NT_CAP_TypeDef *CAPx, CAP_Init_TypeDef *CAP_↔ InitStruct)	491
8.7.4.16	CAP_ITCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_ITSource, FunctionalState State)	492

8.7.4.17	CAP_ITForceCmd(NT_CAP_TypeDef *CAPx, uint32_t CAP_IT← Source) . . . . .	492
8.7.4.18	CAP_ITPendClear(NT_CAP_TypeDef *CAPx) . . . . .	492
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) . . . . .	493
8.7.4.21	CAP_ITStatusClear(NT_CAP_TypeDef *CAPx, uint32_t CAP_IT← Source) . . . . .	493
8.7.4.22	CAP_PWM_GetCompare(NT_CAP_TypeDef *CAPx) . . . . .	494
8.7.4.23	CAP_PWM_GetPeriod(NT_CAP_TypeDef *CAPx) . . . . .	494
8.7.4.24	CAP_PWM_GetShadowCompare(NT_CAP_TypeDef *CAPx) . . . . .	494
8.7.4.25	CAP_PWM_GetShadowPeriod(NT_CAP_TypeDef *CAPx) . . . . .	494
8.7.4.26	CAP_PWM_Init(NT_CAP_TypeDef *CAPx, CAP_PWM_Init_← TypeDef *CAP_PWM_InitStruct) . . . . .	495
8.7.4.27	CAP_PWM_SetCompare(NT_CAP_TypeDef *CAPx, uint32_t ← CompareVal) . . . . .	496
8.7.4.28	CAP_PWM_SetPeriod(NT_CAP_TypeDef *CAPx, uint32_t PeriodVal) . . . . .	496
8.7.4.29	CAP_PWM_SetShadowCompare(NT_CAP_TypeDef *CAPx, uint32_t ← CompareVal) . . . . .	496
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) . . . . .	497
8.7.4.32	CAP_SetShadowTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal) . . . . .	497
8.7.4.33	CAP_SetTimer(NT_CAP_TypeDef *CAPx, uint32_t TimerVal) . . . . .	497
8.7.4.34	CAP_StructInit(CAP_Init_TypeDef *CAP_InitStruct) . . . . .	498
8.7.4.35	CAP_SwSync(NT_CAP_TypeDef *CAPx) . . . . .	498
8.7.4.36	CAP_SyncCmd(NT_CAP_TypeDef *CAPx, FunctionalState State) . . . . .	498
8.7.4.37	CAP_TimerCmd(NT_CAP_TypeDef *CAPx, FunctionalState State) . . . . .	499
8.8	Файл niietcm4_conf.h . . . . .	499
8.8.1	Подробное описание . . . . .	499
8.9	Файл niietcm4_dma.c . . . . .	500
8.9.1	Подробное описание . . . . .	501
8.9.2	Функции . . . . .	501
8.9.2.1	DMA_BasePtrConfig(uint32_t BasePtr) . . . . .	501
8.9.2.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel) . . . . .	502
8.9.2.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State) . . . . .	502
8.9.2.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_← ChannelInit_TypeDef *DMA_ChannelInitStruct) . . . . .	502
8.9.2.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_← ChannelInitStruct) . . . . .	503
8.9.2.6	DMA_ClearErrorStatus() . . . . .	503
8.9.2.7	DMA_DeInit() . . . . .	504

8.9.2.8	DMA_ErrorStatus()	505
8.9.2.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	505
8.9.2.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	505
8.9.2.11	DMA_MasterEnableCmd(FunctionalState State)	506
8.9.2.12	DMA_MasterEnableStatus()	506
8.9.2.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	506
8.9.2.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	506
8.9.2.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	507
8.9.2.16	DMA_StateStatus()	507
8.9.2.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	507
8.9.2.18	DMA_SWRequestCmd(uint32_t DMA_Channel)	508
8.9.2.19	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	508
8.9.2.20	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	508
8.10	Файл niietcm4_dma.h	509
8.10.1	Подробное описание	512
8.10.2	Макросы	513
8.10.2.1	CHANNEL_CFG_CYCLE_CTRL_Msk	513
8.10.2.2	CHANNEL_CFG_CYCLE_CTRL_Pos	513
8.10.2.3	CHANNEL_CFG_DST_INC_Msk	513
8.10.2.4	CHANNEL_CFG_DST_INC_Pos	513
8.10.2.5	CHANNEL_CFG_DST_PROT_CTRL_Msk	513
8.10.2.6	CHANNEL_CFG_DST_PROT_CTRL_Pos	513
8.10.2.7	CHANNEL_CFG_DST_SIZE_Msk	513
8.10.2.8	CHANNEL_CFG_DST_SIZE_Pos	513
8.10.2.9	CHANNEL_CFG_N_MINUS_1_Msk	514
8.10.2.10	CHANNEL_CFG_N_MINUS_1_Pos	514
8.10.2.11	CHANNEL_CFG_NEXT_USEBURST_Msk	514
8.10.2.12	CHANNEL_CFG_NEXT_USEBURST_Pos	514
8.10.2.13	CHANNEL_CFG_R_POWER_Msk	514
8.10.2.14	CHANNEL_CFG_R_POWER_Pos	514
8.10.2.15	CHANNEL_CFG_SRC_INC_Msk	514
8.10.2.16	CHANNEL_CFG_SRC_INC_Pos	514
8.10.2.17	CHANNEL_CFG_SRC_PROT_CTRL_Msk	514
8.10.2.18	CHANNEL_CFG_SRC_PROT_CTRL_Pos	515
8.10.2.19	CHANNEL_CFG_SRC_SIZE_Msk	515
8.10.2.20	CHANNEL_CFG_SRC_SIZE_Pos	515
8.10.2.21	DMA_Channel_0	515
8.10.2.22	DMA_Channel_1	515
8.10.2.23	DMA_Channel_10	515
8.10.2.24	DMA_Channel_11	515

8.10.2.25 DMA_Channel_12	515
8.10.2.26 DMA_Channel_13	515
8.10.2.27 DMA_Channel_14	515
8.10.2.28 DMA_Channel_15	516
8.10.2.29 DMA_Channel_16	516
8.10.2.30 DMA_Channel_17	516
8.10.2.31 DMA_Channel_18	516
8.10.2.32 DMA_Channel_19	516
8.10.2.33 DMA_Channel_2	516
8.10.2.34 DMA_Channel_20	516
8.10.2.35 DMA_Channel_21	516
8.10.2.36 DMA_Channel_22	516
8.10.2.37 DMA_Channel_23	517
8.10.2.38 DMA_Channel_3	517
8.10.2.39 DMA_Channel_4	517
8.10.2.40 DMA_Channel_5	517
8.10.2.41 DMA_Channel_6	517
8.10.2.42 DMA_Channel_7	517
8.10.2.43 DMA_Channel_8	517
8.10.2.44 DMA_Channel_9	517
8.10.2.45 DMA_Channel_ADCSEQ0	517
8.10.2.46 DMA_Channel_ADCSEQ1	517
8.10.2.47 DMA_Channel_ADCSEQ2	518
8.10.2.48 DMA_Channel_ADCSEQ3	518
8.10.2.49 DMA_Channel_ADCSEQ4	518
8.10.2.50 DMA_Channel_ADCSEQ5	518
8.10.2.51 DMA_Channel_ADCSEQ6	518
8.10.2.52 DMA_Channel_ADCSEQ7	518
8.10.2.53 DMA_Channel_All	518
8.10.2.54 DMA_Channel_SPI0_RX	518
8.10.2.55 DMA_Channel_SPI0_TX	518
8.10.2.56 DMA_Channel_SPI1_RX	519
8.10.2.57 DMA_Channel_SPI1_TX	519
8.10.2.58 DMA_Channel_SPI2_RX	519
8.10.2.59 DMA_Channel_SPI2_TX	519
8.10.2.60 DMA_Channel_SPI3_RX	519
8.10.2.61 DMA_Channel_SPI3_TX	519
8.10.2.62 DMA_Channel_UART0_RX	519
8.10.2.63 DMA_Channel_UART0_TX	519
8.10.2.64 DMA_Channel_UART1_RX	519

8.10.2.65 DMA_Channel_UART1_TX . . . . .	519
8.10.2.66 DMA_Channel_UART2_RX . . . . .	520
8.10.2.67 DMA_Channel_UART2_TX . . . . .	520
8.10.2.68 DMA_Channel_UART3_RX . . . . .	520
8.10.2.69 DMA_Channel_UART3_TX . . . . .	520
8.10.2.70 IS_DMA_ARBITRATION_RATE . . . . .	520
8.10.2.71 IS_DMA_DATA_INC . . . . .	520
8.10.2.72 IS_DMA_DATA_SIZE . . . . .	521
8.10.2.73 IS_DMA_MODE . . . . .	521
8.10.2.74 IS_DMA_STATE . . . . .	521
8.10.2.75 IS_GET_DMA_CHANNEL . . . . .	521
8.10.3 Перечисления . . . . .	522
8.10.3.1 DMA_ArbitrationRate_TypeDef . . . . .	522
8.10.3.2 DMA_DataInc_TypeDef . . . . .	522
8.10.3.3 DMA_DataSize_TypeDef . . . . .	523
8.10.3.4 DMA_Mode_TypeDef . . . . .	523
8.10.3.5 DMA_State_TypeDef . . . . .	523
8.10.4 Функции . . . . .	524
8.10.4.1 DMA_BasePtrConfig(uint32_t BasePtr) . . . . .	524
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) . . . . .	525
8.10.4.5 DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct) . . . . .	526
8.10.4.6 DMA_ClearErrorStatus() . . . . .	526
8.10.4.7 DMA_DeInit() . . . . .	527
8.10.4.8 DMA_ErrorStatus() . . . . .	528
8.10.4.9 DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State) . . . . .	528
8.10.4.10 DMA_Init(DMA_Init_TypeDef *DMA_InitStruct) . . . . .	528
8.10.4.11 DMA_MasterEnableCmd(FunctionalState State) . . . . .	529
8.10.4.12 DMA_MasterEnableStatus() . . . . .	529
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) . . . . .	530
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) . . . . .	531
8.10.4.19 DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State) . . . . .	531
8.10.4.20 DMA_WaitOnReqStatus(uint32_t DMA_Channel) . . . . .	531

8.11	Файл <code>niietcm4_extmem.c</code> . . . . .	532
8.11.1	Подробное описание . . . . .	532
8.11.2	Макросы . . . . .	533
8.11.2.1	<code>EXT_MEM_CFG_Reset_Value</code> . . . . .	533
8.11.3	Функции . . . . .	533
8.11.3.1	<code>EXTMEM_DeInit()</code> . . . . .	533
8.11.3.2	<code>EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)</code> . . .	533
8.11.3.3	<code>EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)</code>	533
8.12	Файл <code>niietcm4_extmem.h</code> . . . . .	534
8.12.1	Подробное описание . . . . .	535
8.12.2	Макросы . . . . .	535
8.12.2.1	<code>EXTMEM_CEMask_Addr_11</code> . . . . .	535
8.12.2.2	<code>EXTMEM_CEMask_Addr_11_19</code> . . . . .	536
8.12.2.3	<code>EXTMEM_CEMask_Addr_12</code> . . . . .	536
8.12.2.4	<code>EXTMEM_CEMask_Addr_13</code> . . . . .	536
8.12.2.5	<code>EXTMEM_CEMask_Addr_14</code> . . . . .	536
8.12.2.6	<code>EXTMEM_CEMask_Addr_15</code> . . . . .	536
8.12.2.7	<code>EXTMEM_CEMask_Addr_16</code> . . . . .	536
8.12.2.8	<code>EXTMEM_CEMask_Addr_17</code> . . . . .	536
8.12.2.9	<code>EXTMEM_CEMask_Addr_18</code> . . . . .	536
8.12.2.10	<code>EXTMEM_CEMask_Addr_19</code> . . . . .	536
8.12.2.11	<code>IS_EXTMEM_READ_WAITSTATE</code> . . . . .	536
8.12.2.12	<code>IS_EXTMEM_RW_WAITSTATE</code> . . . . .	537
8.12.2.13	<code>IS_EXTMEM_WIDTH</code> . . . . .	537
8.12.2.14	<code>IS_EXTMEM_WRITE_WAITSTATE</code> . . . . .	537
8.12.3	Перечисления . . . . .	538
8.12.3.1	<code>EXTMEM_ReadWaitState_TypeDef</code> . . . . .	538
8.12.3.2	<code>EXTMEM_RWWaitState_TypeDef</code> . . . . .	538
8.12.3.3	<code>EXTMEM_Width_TypeDef</code> . . . . .	538
8.12.3.4	<code>EXTMEM_WriteWaitState_TypeDef</code> . . . . .	539
8.12.4	Функции . . . . .	539
8.12.4.1	<code>EXTMEM_DeInit()</code> . . . . .	539
8.12.4.2	<code>EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)</code> . . .	539
8.12.4.3	<code>EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)</code>	539
8.13	Файл <code>niietcm4_gpio.c</code> . . . . .	540
8.13.1	Подробное описание . . . . .	541
8.13.2	Макросы . . . . .	542
8.13.2.1	<code>GPIO_DATAOUT_Reset_Value</code> . . . . .	542
8.13.2.2	<code>GPIO_GPIODEN0_Reset_Value</code> . . . . .	542
8.13.2.3	<code>GPIO_GPIODEN1_Reset_Value</code> . . . . .	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 . . . . .	543
8.13.2.7	GPIO_GPIOODSCTLx_Reset_Value . . . . .	543
8.13.2.8	GPIO_GPIOPCTLx_Reset_Value . . . . .	543
8.13.2.9	GPIO_GPIOPUCTLx_Reset_Value . . . . .	543
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 . . . . .	544
8.13.2.15	GPIO_Regs_B_D_F_H_Mask . . . . .	544
8.13.2.16	GPIO_Regs_GPIOA_Mask . . . . .	544
8.13.2.17	GPIO_Regs_GPIOB_Mask . . . . .	544
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 . . . . .	545
8.13.3	Функции . . . . .	545
8.13.3.1	GPIO_AltFuncConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO↵ _Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc) . . . . .	545
8.13.3.2	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	545
8.13.3.3	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx) . . . . .	545
8.13.3.4	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GP↵ IO_InitStruct) . . . . .	546
8.13.3.5	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	546
8.13.3.6	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol) . . . . .	546
8.13.3.7	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO↵ _Pin) . . . . .	547
8.13.3.8	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State) . . . . .	547
8.13.3.9	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod) . . . . .	547
8.13.3.10	GPIO_Read(NT_GPIO_TypeDef *GPIOx) . . . . .	548
8.13.3.11	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	548
8.13.3.12	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal) . . . . .	548
8.13.3.13	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin) . . . . .	549
8.13.3.14	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct) . . . . .	549

8.13.3.15	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	549
8.13.3.16	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	550
8.13.3.17	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	550
8.13.3.18	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	550
8.13.3.19	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	551
8.14	Файл niietcm4_gpio.h	551
8.14.1	Подробное описание	554
8.14.2	Макросы	554
8.14.2.1	GPIO_Pin_0	555
8.14.2.2	GPIO_Pin_0_3	555
8.14.2.3	GPIO_Pin_0_7	555
8.14.2.4	GPIO_Pin_1	555
8.14.2.5	GPIO_Pin_10	555
8.14.2.6	GPIO_Pin_11	555
8.14.2.7	GPIO_Pin_12	555
8.14.2.8	GPIO_Pin_12_15	555
8.14.2.9	GPIO_Pin_13	555
8.14.2.10	GPIO_Pin_14	556
8.14.2.11	GPIO_Pin_15	556
8.14.2.12	GPIO_Pin_2	556
8.14.2.13	GPIO_Pin_3	556
8.14.2.14	GPIO_Pin_4	556
8.14.2.15	GPIO_Pin_4_7	556
8.14.2.16	GPIO_Pin_5	556
8.14.2.17	GPIO_Pin_6	556
8.14.2.18	GPIO_Pin_7	556
8.14.2.19	GPIO_Pin_8	556
8.14.2.20	GPIO_Pin_8_11	557
8.14.2.21	GPIO_Pin_8_15	557
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	558
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	559



8.14.2.31	IS_GPIO_OUT	559
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	560
8.14.2.36	IS_GPIO_SYNC	560
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	561
8.14.3.5	GPIO_IntType_TypeDef	561
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	562
8.14.3.10	GPIO_PullUp_TypeDef	562
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	563
8.14.4.1	GPIO_AltFuncConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc)	563
8.14.4.2	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	564
8.14.4.3	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	564
8.14.4.4	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	564
8.14.4.5	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	565
8.14.4.6	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	565
8.14.4.7	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	566
8.14.4.8	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	566
8.14.4.9	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	566
8.14.4.10	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	567
8.14.4.11	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	567
8.14.4.12	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	567
8.14.4.13	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	568
8.14.4.14	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	568

8.14.4.15	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	568
8.14.4.16	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	569
8.14.4.17	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	569
8.14.4.18	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	569
8.14.4.19	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	570
8.15	Файл niietcm4_rcc.c	570
8.15.1	Подробное описание	571
8.15.2	Макросы	572
8.15.2.1	RCC_PLL_CTRL_Reset_Value	572
8.15.2.2	RCC_PLL_NF_Reset_Value	572
8.15.2.3	RCC_PLL_NR_Reset_Value	572
8.15.2.4	RCC_PLL_OD_Reset_Value	572
8.15.3	Функции	572
8.15.3.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	572
8.15.3.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	573
8.15.3.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	573
8.15.3.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	574
8.15.3.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	574
8.15.3.6	RCC_PLLDeInit()	575
8.15.3.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	575
8.15.3.8	RCC_PLLPowerDownCmd(FunctionalState State)	576
8.15.3.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	576
8.15.3.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	576
8.15.3.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	577
8.15.3.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	577
8.15.3.13	RCC_SysClkDiv2Out(FunctionalState State)	577
8.15.3.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	578
8.15.3.15	RCC_SysClkStatus()	578
8.15.3.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	578
8.15.3.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	579
8.15.3.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	579

8.15.3.19	RCC_USBCLKCmd(FunctionalState State)	579
8.15.3.20	RCC_USBCLKConfig(RCC_USBCLK_TypeDef RCC_USBCLK, RCC_↵ USBFreq_TypeDef RCC_USBFreq)	580
8.15.3.21	RCC_WaitClkChange(RCC_SysClk_TypeDef RCC_SysClk)	580
8.16	Файл niietcm4_rcc.h	580
8.16.1	Подробное описание	583
8.16.2	Макросы	584
8.16.2.1	IS_RCC_ADC_CLK	584
8.16.2.2	IS_RCC_PERIPH_CLK	584
8.16.2.3	IS_RCC_PLL_NO	585
8.16.2.4	IS_RCC_PLL_REF	585
8.16.2.5	IS_RCC_SPI_CLK	585
8.16.2.6	IS_RCC_SYS_CLK	585
8.16.2.7	IS_RCC_UART_CLK	586
8.16.2.8	IS_RCC_USB_CLK	586
8.16.2.9	IS_RCC_USB_FREQ	586
8.16.2.10	RCC_CLK_CHANGE_TIMEOUT	586
8.16.2.11	RCC_CLK_PLL_STABLE_TIMEOUT	586
8.16.3	Перечисления	586
8.16.3.1	RCC_ADCClk_TypeDef	586
8.16.3.2	RCC_PeriphClk_TypeDef	587
8.16.3.3	RCC_PeriphRst_TypeDef	587
8.16.3.4	RCC_PLLNO_TypeDef	588
8.16.3.5	RCC_PLLRef_TypeDef	589
8.16.3.6	RCC_SPIClk_TypeDef	589
8.16.3.7	RCC_SysClk_TypeDef	589
8.16.3.8	RCC_UARTClk_TypeDef	589
8.16.3.9	RCC_USBCLK_TypeDef	590
8.16.3.10	RCC_USBFreq_TypeDef	590
8.16.4	Функции	590
8.16.4.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, Functional↵ State State)	590
8.16.4.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	590
8.16.4.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	591
8.16.4.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	591
8.16.4.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32↵ _t SysFreq)	592
8.16.4.6	RCC_PLLDeInit()	592
8.16.4.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	592

8.16.4.8	RCC_PLLPowerDownCmd(FunctionalState State)	593
8.16.4.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	594
8.16.4.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	594
8.16.4.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	594
8.16.4.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	595
8.16.4.13	RCC_SysClkDiv2Out(FunctionalState State)	595
8.16.4.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	595
8.16.4.15	RCC_SysClkStatus()	596
8.16.4.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	596
8.16.4.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	596
8.16.4.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	597
8.16.4.19	RCC_USBClkCmd(FunctionalState State)	597
8.16.4.20	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	597
8.17	Файл niietcm4_rtc.c	597
8.17.1	Подробное описание	598
8.18	Файл niietcm4_rtc.h	598
8.18.1	Подробное описание	600
8.18.2	Макросы	600
8.18.2.1	IS_RTC_FORMAT	600
8.18.2.2	IS_RTC_MONTH	600
8.18.2.3	IS_RTC_WEEKDAY	601
8.18.3	Перечисления	601
8.18.3.1	RTC_Format_TypeDef	601
8.18.3.2	RTC_Month_TypeDef	601
8.18.3.3	RTC_Weekday_TypeDef	602
8.19	Файл niietcm4_timer.c	602
8.19.1	Подробное описание	603
8.19.2	Функции	603
8.19.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	603
8.19.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	603
8.19.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	604
8.19.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	604
8.19.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	604
8.19.2.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	605

8.19.2.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx) . . . . .	605
8.19.2.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx) . . . . .	605
8.19.2.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_↵ t TimerClkFreq, uint32_t TimerPeriod) . . . . .	606
8.19.2.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_↵ t CounterVal) . . . . .	606
8.19.2.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_↵ t ReloadVal) . . . . .	606
8.20	Файл niyetcm4_timer.h . . . . .	606
8.20.1	Подробное описание . . . . .	607
8.20.2	Макросы . . . . .	608
8.20.2.1	IS_TIMER_EXT_INPUT . . . . .	608
8.20.3	Перечисления . . . . .	608
8.20.3.1	TIMER_ExtInput_TypeDef . . . . .	608
8.20.4	Функции . . . . .	609
8.20.4.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State) . . . . .	609
8.20.4.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_↵ _ExtInput_TypeDef TIMER_ExtInput) . . . . .	610
8.20.4.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_↵ t TimerClkFreq, uint32_t TimerFreq) . . . . .	610
8.20.4.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx) . . . . .	610
8.20.4.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx) . . . . .	611
8.20.4.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State) . . . . .	611
8.20.4.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx) . . . . .	611
8.20.4.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx) . . . . .	611
8.20.4.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_↵ t TimerClkFreq, uint32_t TimerPeriod) . . . . .	612
8.20.4.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_↵ t CounterVal) . . . . .	612
8.20.4.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_↵ t ReloadVal) . . . . .	612
8.21	Файл niyetcm4_uart.c . . . . .	613
8.21.1	Подробное описание . . . . .	614
8.21.2	Функции . . . . .	615
8.21.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_↵ t IntDiv, uint32_t FracDiv) . . . . .	615
8.21.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State) . . . . .	616
8.21.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State) . . . . .	616
8.21.2.4	UART_DeInit(NT_UART_TypeDef *UARTx) . . . . .	616
8.21.2.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, Functional↵ State State) . . . . .	617
8.21.2.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_↵ Typedef UART_Dir, FunctionalState State) . . . . .	617

8.21.2.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, uint32_t UART← _Error)	617
8.21.2.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx, uint32_t U← ART_Error)	618
8.21.2.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, uint32_t UART← _Flag)	618
8.21.2.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *← UART_InitStruct)	618
8.21.2.11	UART_ITCmd(NT_UART_TypeDef *UARTx, uint32_t UART_IT← Source, FunctionalState State)	619
8.21.2.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_← Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIF← OLevel)	619
8.21.2.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, uint32_t U← ART_ITSource)	619
8.21.2.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, uint32_t UAR← T_ITSource)	620
8.21.2.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, uint32_t UA← RT_ITSource)	620
8.21.2.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_← ModemInit_TypeDef *UART_ModemInitStruct)	620
8.21.2.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_← ModemInitStruct)	621
8.21.2.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	621
8.21.2.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	621
8.21.2.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	622
8.22	Файл niyetcm4_uart.h	622
8.22.1	Подробное описание	625
8.22.2	Макросы	625
8.22.2.1	IS_UART_DATA_WIDTH	625
8.22.2.2	IS_UART_DIR	626
8.22.2.3	IS_UART_FIFO_LEVEL	626
8.22.2.4	IS_UART_PARITY_BIT	626
8.22.2.5	IS_UART_STOP_BIT	626
8.22.2.6	UART_Error_All	626
8.22.2.7	UART_Error_Break	627
8.22.2.8	UART_Error_Frame	627
8.22.2.9	UART_Error_Overflow	627
8.22.2.10	UART_Error_Parity	627
8.22.2.11	UART_Flag_All	627
8.22.2.12	UART_Flag_Busy	627
8.22.2.13	UART_Flag_InvCTS	627
8.22.2.14	UART_Flag_InvDCD	627
8.22.2.15	UART_Flag_InvDSR	627

8.22.2.16	UART_Flag_InvRI	627
8.22.2.17	UART_Flag_RxFIFOEmpty	628
8.22.2.18	UART_Flag_RxFIFOFull	628
8.22.2.19	UART_Flag_TxFIFOEmpty	628
8.22.2.20	UART_Flag_TxFIFOFull	628
8.22.2.21	UART_ITSource_All	628
8.22.2.22	UART_ITSource_ChangeCTS	628
8.22.2.23	UART_ITSource_ChangeDCD	628
8.22.2.24	UART_ITSource_ChangeDSR	628
8.22.2.25	UART_ITSource_ChangeRI	628
8.22.2.26	UART_ITSource_ErrorBreak	629
8.22.2.27	UART_ITSource_ErrorFrame	629
8.22.2.28	UART_ITSource_ErrorOverflow	629
8.22.2.29	UART_ITSource_ErrorParity	629
8.22.2.30	UART_ITSource_RecieveTimeout	629
8.22.2.31	UART_ITSource_RxFIFOLevel	629
8.22.2.32	UART_ITSource_TxFIFOLevel	629
8.22.3	Перечисления	629
8.22.3.1	UART_DataWidth_TypeDef	629
8.22.3.2	UART_Dir_Typedef	630
8.22.3.3	UART_FIFOLevel_TypeDef	630
8.22.3.4	UART_ParityBit_TypeDef	630
8.22.3.5	UART_StopBit_TypeDef	630
8.22.4	Функции	630
8.22.4.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	630
8.22.4.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	631
8.22.4.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	631
8.22.4.4	UART_DeInit(NT_UART_TypeDef *UARTx)	631
8.22.4.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	632
8.22.4.6	UART_DMAMCmd(NT_UART_TypeDef *UARTx, UART_Dir_Typedef UART_Dir, FunctionalState State)	632
8.22.4.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, uint32_t UART_Error)	632
8.22.4.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx, uint32_t UART_Error)	633
8.22.4.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, uint32_t UART_Flag)	633
8.22.4.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	633

8.22.4.11	UART_ITCmd(NT_UART_TypeDef *UARTx, uint32_t UART_IT← Source, FunctionalState State) . . . . .	634
8.22.4.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_← Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIF← OLevel) . . . . .	634
8.22.4.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, uint32_t U← ART_ITSource) . . . . .	634
8.22.4.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, uint32_t UAR← T_ITSource) . . . . .	635
8.22.4.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, uint32_t UA← RT_ITSource) . . . . .	635
8.22.4.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_← ModemInit_TypeDef *UART_ModemInitStruct) . . . . .	635
8.22.4.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_← ModemInitStruct) . . . . .	636
8.22.4.18	UART_RecieveData(NT_UART_TypeDef *UARTx) . . . . .	636
8.22.4.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data) . . .	636
8.22.4.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct) . . . . .	637
8.23	Файл niietcm4_userflash.c . . . . .	637
8.23.1	Подробное описание . . . . .	638
8.23.2	Функции . . . . .	638
8.23.2.1	USERFLASH_FullErase() . . . . .	638
8.23.2.2	USERFLASH_Info_PageErase(uint32_t PageNum) . . . . .	638
8.23.2.3	USERFLASH_Info_Read(uint32_t Address) . . . . .	639
8.23.2.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data) . . . . .	639
8.23.2.5	USERFLASH_Init(uint32_t SysClkFreq) . . . . .	639
8.23.2.6	USERFLASH_ITCmd(FunctionalState State) . . . . .	639
8.23.2.7	USERFLASH_OperationStatus() . . . . .	640
8.23.2.8	USERFLASH_OperationStatusClear() . . . . .	640
8.23.2.9	USERFLASH_PageErase(uint32_t PageNum) . . . . .	640
8.23.2.10	USERFLASH_Read(uint32_t Address) . . . . .	640
8.23.2.11	USERFLASH_Write(uint32_t Address, uint32_t Data) . . . . .	641
8.24	Файл niietcm4_userflash.h . . . . .	641
8.24.1	Подробное описание . . . . .	642
8.24.2	Макросы . . . . .	643
8.24.2.1	IS_USERFLASH_STATUS . . . . .	643
8.24.2.2	USERFLASH_INFO_PAGE_SIZE_BYTES . . . . .	643
8.24.2.3	USERFLASH_INFO_PAGE_TOTAL . . . . .	643
8.24.2.4	USERFLASH_INFO_TOTAL_BYTES . . . . .	643
8.24.2.5	USERFLASH_PAGE_SIZE_BYTES . . . . .	643
8.24.2.6	USERFLASH_PAGE_TOTAL . . . . .	644
8.24.2.7	USERFLASH_TOTAL_BYTES . . . . .	644



8.24.3	Перечисления	644
8.24.3.1	USERFLASH_Status_TypeDef	644
8.24.4	Функции	644
8.24.4.1	USERFLASH_FullErase()	644
8.24.4.2	USERFLASH_Info_PageErase(uint32_t PageNum)	644
8.24.4.3	USERFLASH_Info_Read(uint32_t Address)	645
8.24.4.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	645
8.24.4.5	USERFLASH_Init(uint32_t SysClkFreq)	645
8.24.4.6	USERFLASH_ITCmd(FunctionalState State)	645
8.24.4.7	USERFLASH_OperationStatus()	646
8.24.4.8	USERFLASH_OperationStatusClear()	646
8.24.4.9	USERFLASH_PageErase(uint32_t PageNum)	646
8.24.4.10	USERFLASH_Read(uint32_t Address)	646
8.24.4.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	647
8.25	Файл niietcm4_watchdog.c	647
8.25.1	Подробное описание	648
8.25.2	Макросы	648
8.25.2.1	WATCHDOG_Lock_Value	648
8.25.2.2	WATCHDOG_Unlock_Value	648
8.25.3	Функции	649
8.25.3.1	WATCHDOG_Cmd(FunctionalState State)	649
8.25.3.2	WATCHDOG_GetCounter()	649
8.25.3.3	WATCHDOG_GetReload()	649
8.25.3.4	WATCHDOG_ITMaskedStatus()	649
8.25.3.5	WATCHDOG_ITRawStatus()	649
8.25.3.6	WATCHDOG_ITStatusClear()	649
8.25.3.7	WATCHDOG_LockCmd(FunctionalState State)	650
8.25.3.8	WATCHDOG_RstCmd(FunctionalState State)	650
8.25.3.9	WATCHDOG_SetReload(uint32_t ReloadVal)	650
8.26	Файл niietcm4_watchdog.h	650
8.26.1	Подробное описание	651
8.26.2	Функции	652
8.26.2.1	WATCHDOG_Cmd(FunctionalState State)	652
8.26.2.2	WATCHDOG_GetCounter()	652
8.26.2.3	WATCHDOG_GetReload()	652
8.26.2.4	WATCHDOG_ITMaskedStatus()	652
8.26.2.5	WATCHDOG_ITRawStatus()	653
8.26.2.6	WATCHDOG_ITStatusClear()	653
8.26.2.7	WATCHDOG_LockCmd(FunctionalState State)	653
8.26.2.8	WATCHDOG_RstCmd(FunctionalState State)	653

8.26.2.9 WATCHDOG_SetReload(uint32_t ReloadVal) . . . . .	654
Алфавитный указатель . . . . .	657

# 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
- 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 v0.9.0](#). 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](mailto:kolbov@niiet.ru);

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

## Глава 2

# Информация о релизах

### v0.9.0

Реорганизация библиотеки:

- изменена структура каталогов, теперь она соответствует рекомендациям CMSIS
- добавлены все необходимые файлы из CMSIS v.4.5.0
- примеры и шаблоны были обновлены, чтобы корректно работать с новым расположением каталогов
- обновлен хедер периферии
- устранено дублирование хедера периферии и скриптов линкера в примерах
- имена регистров и полей, используемых внутри библиотеки, были приведены к новому хедеру

Общие изменения:

- добавлена документация функции [GPIO\\_AltFuncConfig\(\)](#)
- улучшены механизмы работы с флагами UART, CAP

Фиксы:

- ошибки инициализации uart для printf в примерах DMA
- потери входящих данных при использовании [UART\\_SendData\(\)](#)

### 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
- не критичные 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 . . . . .	363
Настройка драйвера . . . . .	364
Макросы . . . . .	365
Типы . . . . .	366
Периферия . . . . .	369
ADC . . . . .	236
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 . . . . .	237
Приватные константы . . . . .	238
Начальные значения регистров . . . . .	239
Приватные функции . . . . .	240
BOOTFLASH . . . . .	256
Константы . . . . .	62
Основная область флеш . . . . .	63
Информационная область флеш . . . . .	64
Типы . . . . .	65
Функции . . . . .	66
Основная область флеш . . . . .	68
Информационная область флеш . . . . .	70
Приватные данные . . . . .	257
Приватные константы . . . . .	258
Приватные функции . . . . .	259
CAP . . . . .	262
Типы . . . . .	71

Константы . . . . .	75
Маски источников прерываний . . . . .	76
Функции . . . . .	78
Конфигурация . . . . .	79
Режим ШИМ . . . . .	84
Режим захвата . . . . .	89
Прерывания . . . . .	94
Приватные данные . . . . .	263
Приватные константы . . . . .	264
Приватные функции . . . . .	265
DMA . . . . .	279
Константы . . . . .	97
Маски для CHANNEL_CFG . . . . .	98
Маски каналов DMA . . . . .	101
Маски каналов по номеру . . . . .	102
Маски каналов по имени . . . . .	106
Типы . . . . .	110
Функции . . . . .	115
Инициализация каналов DMA . . . . .	116
Инициализация контроллера DMA . . . . .	118
Конфигурация контроллера DMA . . . . .	120
Статусная информация . . . . .	124
Приватные данные . . . . .	280
Приватные константы . . . . .	281
Начальные значения регистров . . . . .	282
Приватные функции . . . . .	283
EXTMEM . . . . .	291
Константы . . . . .	126
Маски адреса . . . . .	127
Типы . . . . .	129
Функции . . . . .	133
Приватные данные . . . . .	292
Приватные константы . . . . .	293
Начальные значения регистров . . . . .	294
Приватные функции . . . . .	295
GPIO . . . . .	297
Типы . . . . .	135
Константы . . . . .	142
Маски пинов . . . . .	143
Функции . . . . .	147
Инициализация и деинициализация . . . . .	148
Чтение и запись . . . . .	150
Чтение . . . . .	151
Запись . . . . .	153
Битовые операции . . . . .	155
Фильтрация . . . . .	157
Прерывания . . . . .	159
Приватные данные . . . . .	298
Приватные константы . . . . .	299
Начальные значения регистров . . . . .	300
Маски портов . . . . .	303
Приватные функции . . . . .	305
RCC . . . . .	314
Константы . . . . .	161
Типы . . . . .	162
Функции . . . . .	170

Конфигурация PLL . . . . .	171
Управление тактированием . . . . .	174
Тактирование USB . . . . .	176
Тактирование UART . . . . .	177
Тактирование SPI . . . . .	180
Тактирование ADC . . . . .	182
Управление сбросом . . . . .	183
Приватные данные . . . . .	315
Приватные константы . . . . .	316
Начальные значения регистров . . . . .	317
Приватные функции . . . . .	318
RTC . . . . .	327
Типы . . . . .	184
Функции . . . . .	187
Приватные данные . . . . .	328
Приватные функции . . . . .	329
TIMER . . . . .	330
Типы . . . . .	188
Константы . . . . .	189
Функции . . . . .	190
Конфигурация . . . . .	191
Прерывания . . . . .	195
Приватные данные . . . . .	331
Приватные константы . . . . .	332
Приватные функции . . . . .	333
UART . . . . .	337
Типы . . . . .	197
Константы . . . . .	201
Источники прерываний UART . . . . .	202
Флаги работы UART . . . . .	204
Ошибки приемника UART . . . . .	206
Функции . . . . .	207
Инициализация и деинициализация . . . . .	209
Прием и передача . . . . .	212
Режим модема . . . . .	215
Прерывания . . . . .	216
Настройка DMA . . . . .	219
Приватные данные . . . . .	338
Приватные константы . . . . .	339
Приватные функции . . . . .	340
USERFLASH . . . . .	350
Константы . . . . .	220
Основная область флеш . . . . .	221
Информационная область флеш . . . . .	222
Типы . . . . .	223
Функции . . . . .	224
Основная область флеш . . . . .	226
Информационная область флеш . . . . .	228
Приватные данные . . . . .	351
Приватные константы . . . . .	352
Приватные функции . . . . .	353
WATCHDOG . . . . .	357
Типы . . . . .	230
Константы . . . . .	231
Функции . . . . .	232
Конфигурация . . . . .	233

Прерывания . . . . .	235
Приватные данные . . . . .	358
Приватные константы . . . . .	359
Приватные функции . . . . .	360

## Глава 4

# Алфавитный указатель структур данных

### 4.1 Структуры данных

Структуры данных с их кратким описанием.

<a href="#">_CHANNEL_CFG_bits</a>	Битовый доступ к регистру CHANNEL_CFG в <a href="#">DMA_Channel_TypeDef</a> . . . . .	371
<a href="#">ADC_DC_Init_TypeDef</a>	Структура инициализации цифровых компараторов . . . . .	373
<a href="#">ADC_Init_TypeDef</a>	Структура инициализации модулей АЦП . . . . .	375
<a href="#">ADC_SEQ_Init_TypeDef</a>	Структура инициализации секвенсоров . . . . .	376
<a href="#">CAP_Capture_Init_TypeDef</a>	Структура инициализации режима захвата . . . . .	377
<a href="#">CAP_Init_TypeDef</a>	Структура инициализации блока захвата в целом . . . . .	380
<a href="#">CAP_PWM_Init_TypeDef</a>	Структура инициализации режима ШИМ . . . . .	381
<a href="#">DMA_Channel_TypeDef</a>	Тип, описывающий структуру канала DMA . . . . .	382
<a href="#">DMA_ChannelInit_TypeDef</a>	Структура инициализации канала DMA . . . . .	383
<a href="#">DMA_ConfigData_TypeDef</a>	Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ . . . . .	385
<a href="#">DMA_ConfigStruct_TypeDef</a>	Управляющая структура данных DMA . . . . .	386
<a href="#">DMA_Init_TypeDef</a>	Структура инициализации контроллера DMA . . . . .	387
<a href="#">DMA_Protect_TypeDef</a>	Защита шины при чтении из источника или записи в приемник через DMA . . . . .	388
<a href="#">EXTMEM_Init_TypeDef</a>	Структура инициализации внешней памяти . . . . .	389
<a href="#">GPIO_Init_TypeDef</a>	Структура инициализации GPIO . . . . .	390
<a href="#">RCC_PLLInit_TypeDef</a>	Структура инициализации PLL . . . . .	392
<a href="#">RTC_Date_TypeDef</a>	Структура даты . . . . .	393
<a href="#">RTC_Time_TypeDef</a>	Структура времени . . . . .	394
<a href="#">UART_Init_TypeDef</a>	Структура инициализации UART . . . . .	395

<a href="#">UART_ModemInit_TypeDef</a>	
Структура инициализации модемного режима . . . . .	<a href="#">397</a>

## Глава 5

# Список файлов

### 5.1 Файлы

Полный список документированных файлов.

<a href="#">niietcm4.h</a>	Это главный заголовочный файл драйвера, обычно включаемый в main.c . . . . .	399
<a href="#">niietcm4_adc.c</a>	Файл содержит реализацию всех функции для работы с модулями АЦП, секвенсорами, цифровыми компараторами . . . . .	403
<a href="#">niietcm4_adc.h</a>	Файл содержит все прототипы функций для работы с АЦП, секвенсорами, цифровыми компараторами . . . . .	420
<a href="#">niietcm4_bootflash.c</a>	Файл содержит реализацию всех функции для работы с загрузочной флеш . . . . .	458
<a href="#">niietcm4_bootflash.h</a>	Файл содержит все прототипы функций для загрузочной флеш . . . . .	461
<a href="#">niietcm4_cap.c</a>	Файл содержит реализацию всех функции для работы с блоками захвата . . . . .	466
<a href="#">niietcm4_cap.h</a>	Файл содержит все прототипы функций для блоков захвата . . . . .	480
<a href="#">niietcm4_conf.h</a>	Файл конфигурации драйвера . . . . .	499
<a href="#">niietcm4_dma.c</a>	Файл содержит реализацию всех функции для работы с DMA . . . . .	500
<a href="#">niietcm4_dma.h</a>	Файл содержит все прототипы функций для DMA . . . . .	509
<a href="#">niietcm4_extmem.c</a>	Файл содержит реализацию всех функции для работы с интерфейсом внешней памяти . . . . .	532
<a href="#">niietcm4_extmem.h</a>	Файл содержит все прототипы функций для интерфейса внешней памяти . . . . .	534
<a href="#">niietcm4_gpio.c</a>	Файл содержит реализацию всех функции для работы с модулями GPIO . . . . .	540
<a href="#">niietcm4_gpio.h</a>	Файл содержит все прототипы функций для GPIO . . . . .	551
<a href="#">niietcm4_rcc.c</a>	Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера . . . . .	570
<a href="#">niietcm4_rcc.h</a>	Файл содержит все прототипы функций для RCC (Reset & Clock Control) . . . . .	580
<a href="#">niietcm4_rtc.c</a>	Файл содержит реализацию всех функции для работы с RTC . . . . .	597

<a href="#">niietcm4_rtc.h</a>	Файл содержит все прототипы функций для таймеров . . . . .	598
<a href="#">niietcm4_timer.c</a>	Файл содержит реализацию всех функции для работы с таймерами . . . . .	602
<a href="#">niietcm4_timer.h</a>	Файл содержит все прототипы функций для таймеров . . . . .	606
<a href="#">niietcm4_uart.c</a>	Файл содержит реализацию всех функции для работы с модулями UART . . . .	613
<a href="#">niietcm4_uart.h</a>	Файл содержит все прототипы функций для UART . . . . .	622
<a href="#">niietcm4_userflash.c</a>	Файл содержит реализацию всех функции для работы с пользовательской флеш .	637
<a href="#">niietcm4_userflash.h</a>	Файл содержит все прототипы функций для пользовательской флеш . . . . .	641
<a href="#">niietcm4_watchdog.c</a>	Файл содержит реализацию всех функции для работы со сторожевым таймером .	647
<a href="#">niietcm4_watchdog.h</a>	Файл содержит все прототипы функций для сторожевого таймера . . . . .	650



## 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 Маски выбора секвенсоров

### Макросы

- `#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)0x<←  
FFFFFF00) == ((uint32_t)0x00)))`

Макрос проверки попадания масок секвенсоров в допустимый диапазон.

### 6.4.1 Подробное описание

### 6.4.2 Макросы

#### 6.4.2.1 `#define ADC_SEQ_0 ((uint32_t)0x00000001)`

Секвенсор 0

См. определение в файле `niietcm4_adc.h` строка 137

#### 6.4.2.2 `#define ADC_SEQ_1 ((uint32_t)0x00000002)`

Секвенсор 1

См. определение в файле `niietcm4_adc.h` строка 138

#### 6.4.2.3 `#define ADC_SEQ_2 ((uint32_t)0x00000004)`

Секвенсор 2

См. определение в файле `niietcm4_adc.h` строка 139

#### 6.4.2.4 `#define ADC_SEQ_3 ((uint32_t)0x00000008)`

Секвенсор 3

См. определение в файле `niietcm4_adc.h` строка 140

#### 6.4.2.5 `#define ADC_SEQ_4 ((uint32_t)0x00000010)`

Секвенсор 4

См. определение в файле `niietcm4_adc.h` строка 141

#### 6.4.2.6 `#define ADC_SEQ_5 ((uint32_t)0x00000020)`

Секвенсор 5

См. определение в файле `niietcm4_adc.h` строка 142

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 <a href="#">ADC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
--------------------	---

## 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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
--------------------	---



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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	---

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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_Module_TypeDef</a> .
------------	---

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 <a href="#">ADC_Module_TypeDef</a> .
ADC_InitStruct	Указатель на структуру типа <a href="#">ADC_Init_TypeDef</a> , 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 <a href="#">ADC_Init_TypeDef</a> , 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 [ADC\\_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

<a href="#">ADC_DC_↔ Module</a>	Выбор модуля цифрового компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Module_TypeDef</a> .
-------------------------------------	--

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 [ADC\\_DC\\_InitStruct](#).

Аргументы

<a href="#">ADC_DC_↔ Module</a>	Выбор модуля цифрового компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Module_TypeDef</a> .
<a href="#">ADC_DC_↔ InitStruct</a>	Указатель на структуру типа <a href="#">ADC_DC_Init_TypeDef</a> , 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](#).

##### 6.9.2.3 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 <a href="#">ADC_DC_Init_TypeDef</a> , 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
---------------------	---

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](#).

#### 6.10.2.2 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 <a href="#">ADC_SEQ_Init_TypeDef</a> .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа <a href="#">ADC_SEQ_Init_TypeDef</a> , 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](#).

6.10.2.3 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 <a href="#">ADC_SEQ_Init_TypeDef</a> , 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::ADC\_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\_SEQ\_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

<a href="#">ADC_SEQ_↔ Module</a>	Выбор секвенсора. Параметр принимает любое значение из <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

#### 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

<a href="#">ADC_SEQ_↔ Module</a>	Выбор секвенсора. Параметр принимает любое значение из <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
<a href="#">ADC_SEQ_↔ FIFOLevel</a>	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из <a href="#">ADC_SEQ_FIFOLevel_TypeDef</a> .

#### 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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

<a href="#">ADC_DC_Mode_TypeDef</a>	Выбор цифрового компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

#### 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 <a href="#">ADC_DC_Module_TypeDef</a> .
ADC_DC_↔ Mode	Режим срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Mode_TypeDef</a> .
ADC_DC_↔ Condition	Условие срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Condition_TypeDef</a> .

## 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">AD↔C_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">AD↔C_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">AD↔C_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
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 <a href="#">ADC_SEQ_IT↔ CountRst</a> . Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 флеш.
- void [BOOTFLASH\\_OperationStatusClear](#) ()  
Очищает статус работы контроллера загрузочной флеш.
- 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

<a href="#">SysClkFreq</a>	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

<a href="#">State</a>	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-----------------------	---

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 флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из <a href="#">BOOTFLASH_H_Status_TypeDef</a> .
--------	---

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) & ~CAP_ITSource_All) == 0)`

Макрос проверки источников прерываний на попадание в допустимый диапазон.

### 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

## 6.25 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 <a href="#">FunctionalState</a> .

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 <a href="#">CAP_PWM_Init_TypeDef</a> , 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 <a href="#">FunctionalState</a> .

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 <a href="#">CAP_Capture_Init_TypeDef</a> , 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

CAP_↵ Capture_Init↵ Struct	Указатель на структуру типа <a href="#">CAP_Capture_Init_TypeDef</a> , 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 <a href="#">Маски источников прерываний</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">Маски источников прерываний</a> .

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 681

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 657

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 <a href="#">Маски источников прерываний</a> .

Возвращаемые значения

Status	Статус прерывания. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_cap.c строка 616

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_CAP\_IT\_SOURCE.

6.29.2.6 void CAP\_ITStatusClear ( NT\_CAP\_TypeDef \* CAPx, uint32\_t CAP\_ITSource )

Сброс флагов источников прерываний выбранного блока захвата.

## Аргументы

CAP <sub>x</sub>	Выбор CAP, где x лежит в диапазоне 0-5.
CAP_IT↔ Source	Выбор источников прерывания. Параметр принимает любую совокупность значений из <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_cap.c строка 643

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

### 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)) || \
  ((CHANNEL) == (DMA_Channel_1)) || \
  ((CHANNEL) == (DMA_Channel_2)) || \
  ((CHANNEL) == (DMA_Channel_3)) || \
  ((CHANNEL) == (DMA_Channel_4)) || \
  ((CHANNEL) == (DMA_Channel_5)) || \
  ((CHANNEL) == (DMA_Channel_6)) || \
  ((CHANNEL) == (DMA_Channel_7)) || \
  ((CHANNEL) == (DMA_Channel_8)) || \
  ((CHANNEL) == (DMA_Channel_9)) || \
  ((CHANNEL) == (DMA_Channel_10)) || \
  ((CHANNEL) == (DMA_Channel_11)) || \
  ((CHANNEL) == (DMA_Channel_12)) || \
  ((CHANNEL) == (DMA_Channel_13)) || \
  ((CHANNEL) == (DMA_Channel_14)) || \
  ((CHANNEL) == (DMA_Channel_15)) || \
  ((CHANNEL) == (DMA_Channel_16)) || \
  ((CHANNEL) == (DMA_Channel_17)) || \
  ((CHANNEL) == (DMA_Channel_18)) || \
  ((CHANNEL) == (DMA_Channel_19)) || \
  ((CHANNEL) == (DMA_Channel_20)) || \
  ((CHANNEL) == (DMA_Channel_21)) || \
  ((CHANNEL) == (DMA_Channel_22)) || \
  ((CHANNEL) == (DMA_Channel_23)))
```

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 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)`

#### 6.33.1 Подробное описание

#### 6.33.2 Макросы

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

Канал DMA 0

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

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

Канал DMA 1

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

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

Канал DMA 10

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

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

Канал DMA 11

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



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

Канал DMA 12

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

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

Канал DMA 13

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

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

Канал DMA 14

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

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

Канал DMA 15

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

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

Канал DMA 16

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

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

Канал DMA 17

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

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

Канал DMA 18

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

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

Канал DMA 19

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

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

Канал DMA 2

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

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

Канал DMA 20

См. определение в файле niietcm4\_dma.h строка 113

6.33.2.15 #define DMA\_Channel\_21 ((uint32\_t)0x00200000)

Канал DMA 21

См. определение в файле niietcm4\_dma.h строка 114

6.33.2.16 #define DMA\_Channel\_22 ((uint32\_t)0x00400000)

Канал DMA 22

См. определение в файле niietcm4\_dma.h строка 115

6.33.2.17 #define DMA\_Channel\_23 ((uint32\_t)0x00800000)

Канал DMA 23

См. определение в файле niietcm4\_dma.h строка 116

6.33.2.18 #define DMA\_Channel\_3 ((uint32\_t)0x00000008)

Канал DMA 3

См. определение в файле niietcm4\_dma.h строка 96

6.33.2.19 #define DMA\_Channel\_4 ((uint32\_t)0x00000010)

Канал DMA 4

См. определение в файле niietcm4\_dma.h строка 97

6.33.2.20 #define DMA\_Channel\_5 ((uint32\_t)0x00000020)

Канал DMA 5

См. определение в файле niietcm4\_dma.h строка 98

6.33.2.21 #define DMA\_Channel\_6 ((uint32\_t)0x00000040)

Канал DMA 6

См. определение в файле niietcm4\_dma.h строка 99

6.33.2.22 #define DMA\_Channel\_7 ((uint32\_t)0x00000080)

Канал DMA 7

См. определение в файле niietcm4\_dma.h строка 100

6.33.2.23 #define DMA\_Channel\_8 ((uint32\_t)0x00000100)

Канал DMA 8

См. определение в файле niietcm4\_dma.h строка 101

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

Канал DMA 9

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

## 6.34 Маски каналов по имени

### Макросы

- `#define DMA_Channel_UART0_TX DMA_Channel_0`
- `#define DMA_Channel_UART1_TX DMA_Channel_1`
- `#define DMA_Channel_UART2_TX DMA_Channel_2`
- `#define DMA_Channel_UART3_TX DMA_Channel_3`
- `#define DMA_Channel_UART0_RX DMA_Channel_4`
- `#define DMA_Channel_UART1_RX DMA_Channel_5`
- `#define DMA_Channel_UART2_RX DMA_Channel_6`
- `#define DMA_Channel_UART3_RX DMA_Channel_7`
- `#define DMA_Channel_ADCSEQ0 DMA_Channel_8`
- `#define DMA_Channel_ADCSEQ1 DMA_Channel_9`
- `#define DMA_Channel_ADCSEQ2 DMA_Channel_10`
- `#define DMA_Channel_ADCSEQ3 DMA_Channel_11`
- `#define DMA_Channel_ADCSEQ4 DMA_Channel_12`
- `#define DMA_Channel_ADCSEQ5 DMA_Channel_13`
- `#define DMA_Channel_ADCSEQ6 DMA_Channel_14`
- `#define DMA_Channel_ADCSEQ7 DMA_Channel_15`
- `#define DMA_Channel_SPI0_TX DMA_Channel_16`
- `#define DMA_Channel_SPI1_TX DMA_Channel_17`
- `#define DMA_Channel_SPI2_TX DMA_Channel_18`
- `#define DMA_Channel_SPI3_TX DMA_Channel_19`
- `#define DMA_Channel_SPI0_RX DMA_Channel_20`
- `#define DMA_Channel_SPI1_RX DMA_Channel_21`
- `#define DMA_Channel_SPI2_RX DMA_Channel_22`
- `#define DMA_Channel_SPI3_RX DMA_Channel_23`

#### 6.34.1 Подробное описание

#### 6.34.2 Макросы

##### 6.34.2.1 `#define DMA_Channel_ADCSEQ0 DMA_Channel_8`

Канал DMA секвенсора 0 АЦП

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

##### 6.34.2.2 `#define DMA_Channel_ADCSEQ1 DMA_Channel_9`

Канал DMA секвенсора 1 АЦП

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

##### 6.34.2.3 `#define DMA_Channel_ADCSEQ2 DMA_Channel_10`

Канал DMA секвенсора 2 АЦП

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

##### 6.34.2.4 `#define DMA_Channel_ADCSEQ3 DMA_Channel_11`

Канал DMA секвенсора 3 АЦП

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

6.34.2.5 `#define DMA_Channel_ADCSEQ4 DMA_Channel_12`

Канал DMA секвенсора 4 АЦП

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

6.34.2.6 `#define DMA_Channel_ADCSEQ5 DMA_Channel_13`

Канал DMA секвенсора 5 АЦП

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

6.34.2.7 `#define DMA_Channel_ADCSEQ6 DMA_Channel_14`

Канал DMA секвенсора 6 АЦП

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

6.34.2.8 `#define DMA_Channel_ADCSEQ7 DMA_Channel_15`

Канал DMA секвенсора 7 АЦП

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

6.34.2.9 `#define DMA_Channel_SPI0_RX DMA_Channel_20`

Канал DMA по приему от SPI0

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

6.34.2.10 `#define DMA_Channel_SPI0_TX DMA_Channel_16`

Канал DMA по передаче от SPI0

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

6.34.2.11 `#define DMA_Channel_SPI1_RX DMA_Channel_21`

Канал DMA по приему от SPI1

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

6.34.2.12 `#define DMA_Channel_SPI1_TX DMA_Channel_17`

Канал DMA по передаче от SPI1

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

6.34.2.13 `#define DMA_Channel_SPI2_RX DMA_Channel_22`

Канал DMA по приему от SPI2

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

6.34.2.14 `#define DMA_Channel_SPI2_TX DMA_Channel_18`

Канал DMA по передаче от SPI2

См. определение в файле niietcm4\_dma.h строка 144

6.34.2.15 `#define DMA_Channel_SPI3_RX DMA_Channel_23`

Канал DMA по приему от SPI3

См. определение в файле niietcm4\_dma.h строка 149

6.34.2.16 `#define DMA_Channel_SPI3_TX DMA_Channel_19`

Канал DMA по передаче от SPI3

См. определение в файле niietcm4\_dma.h строка 145

6.34.2.17 `#define DMA_Channel_UART0_RX DMA_Channel_4`

Канал DMA по приему от UART0

См. определение в файле niietcm4\_dma.h строка 130

6.34.2.18 `#define DMA_Channel_UART0_TX DMA_Channel_0`

Канал DMA по передаче от UART0

См. определение в файле niietcm4\_dma.h строка 126

6.34.2.19 `#define DMA_Channel_UART1_RX DMA_Channel_5`

Канал DMA по приему от UART1

См. определение в файле niietcm4\_dma.h строка 131

6.34.2.20 `#define DMA_Channel_UART1_TX DMA_Channel_1`

Канал DMA по передаче от UART1

См. определение в файле niietcm4\_dma.h строка 127

6.34.2.21 `#define DMA_Channel_UART2_RX DMA_Channel_6`

Канал DMA по приему от UART2

См. определение в файле niietcm4\_dma.h строка 132

6.34.2.22 `#define DMA_Channel_UART2_TX DMA_Channel_2`

Канал DMA по передаче от UART2

См. определение в файле niietcm4\_dma.h строка 128

6.34.2.23 `#define DMA_Channel_UART3_RX DMA_Channel_7`

Канал DMA по приему от UART3

См. определение в файле niietcm4\_dma.h строка 133

6.34.2.24 `#define DMA_Channel_UART3_TX DMA_Channel_3`

Канал DMA по передаче от UART3

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

## 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

`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

#### 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



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 <a href="#">DMA_ChannelInit_TypeDef</a> , 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 <a href="#">DMA_Init_TypeDef</a> , 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 <a href="#">DMA_Init_TypeDef</a> , 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">Маски каналов по номеру</a> .
-------------	--

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">DMA_Protect_TypeDef</a> .
---------------------	--

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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">Маски каналов по номеру</a> .
-------------	--

## 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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"> <li>• ОК - ошибок не было;</li> <li>• ERROR - произошла ошибка.</li> </ul>
--------	---

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

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

<a href="#">EXTMEM_InitStruct</a>	Указатель на структуру типа <a href="#">EXTMEM_Init_TypeDef</a> , 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

<a href="#">EXTMEM_InitStruct</a>	Указатель на структуру типа <a href="#">EXTMEM_Init_TypeDef</a> , 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_AltFuncConfig()` и `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 8mA.
- GPIO\_Load\_16mA Максимальный ток 16mA.

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 niietcm4\_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 niietcm4\_gpio.h строка 335

6.47.2.3 #define GPIO\_Pin\_0\_7 ((uint32\_t)0x00FF)

Пины 0-7 выбраны.

См. определение в файле niietcm4\_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\_InitTypeDef** \*GPIO\_InitStruct)  
Инициализирует модуль GPIOx согласно параметрам структуры GPIO\_InitStruct.
- void **GPIO\_StructInit** (**GPIO\_InitTypeDef** \*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)  
Осуществляет выбор альтернативной функции вывода GPIOx.

### 6.49.1 Подробное описание

### 6.49.2 Функции

6.49.2.1 void **GPIO\_AltFuncConfig** ( NT\_GPIO\_TypeDef \* GPIOx, uint32\_t GPIO\_Pin, GPIO\_AltFunc\_TypeDef GPIO\_AltFunc )

Осуществляет выбор альтернативной функции вывода GPIOx.

#### 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.
GPIO_AltFunc	Выбор номера альтернативной функции пина. Параметр принимает любое значение из <b>GPIO_AltFunc_TypeDef</b> .

#### Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 278

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH, IS\_GPIO\_ALT\_FUNC и IS\_GPIO\_PIN.

Используется в GPIO\_Init().

6.49.2.2 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 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.

6.49.2.3 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 <a href="#">GPIO_Init_TypeDef</a> , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 340

Перекрестные ссылки GPIO\_Init\_TypeDef::GPIO\_AltFunc, GPIO\_AltFuncConfig(), 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().

6.49.2.4 void GPIO\_StructInit ( GPIO\_Init\_TypeDef \* GPIO\_InitStruct )

Заполнение каждого члена структуры GPIO\_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа <a href="#">GPIO_Init_TypeDef</a> , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 465

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

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне A..H.
--------------------	--

Возвращаемые значения

Состояние порта <code>GPIOx</code>	
------------------------------------	--

См. определение в файле `niietcm4_gpio.c` строка 512

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

<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

Состояние выбранного пина	
---------------------------	--

См. определение в файле `niietcm4_gpio.c` строка 486

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 527

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 579

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 <a href="#">BitAction</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 555

Перекрестные ссылки Bit\_CLEAR, Bit\_SET, IS\_GET\_GPIO\_PIN, IS\_GPIO\_ALL\_PERIPH и IS\_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 595

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 635

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 618

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 652

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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niitcm4\_gpio.c строка 762

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 <a href="#">GPIO_QualMode_TypeDef</a> .
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 673

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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 823

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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niitcm4\_gpio.c строка 922

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 <a href="#">GPIO_IntType_TypeDef</a> .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из <a href="#">GPIO_IntPol_TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niitcm4\_gpio.c строка 885

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 niietcm4\_gpio.c строка 947

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)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)(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_PL1) || \
    ((SYS_CLK) == RCC_SysClk_PL1DIV) || \
    ((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



<code>RCC_PeriphClk_PWM3</code>	Управление тактированием блока PWM 3
<code>RCC_PeriphClk_PWM4</code>	Управление тактированием блока PWM 4
<code>RCC_PeriphClk_PWM5</code>	Управление тактированием блока PWM 5
<code>RCC_PeriphClk_PWM6</code>	Управление тактированием блока PWM 6
<code>RCC_PeriphClk_PWM7</code>	Управление тактированием блока PWM 7
<code>RCC_PeriphClk_PWM8</code>	Управление тактированием блока PWM 8
<code>RCC_PeriphClk_WD</code>	Управление тактированием сторожевого таймера
<code>RCC_PeriphClk_I2C0</code>	Управление тактированием блока I2C 0
<code>RCC_PeriphClk_I2C1</code>	Управление тактированием блока I2C 1
<code>RCC_PeriphClk_ADC</code>	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

<code>RCC_PeriphRst_WD</code>	Управление сбросом сторожевого таймера
<code>RCC_PeriphRst_I2C0</code>	Управление сбросом блока I2C 0
<code>RCC_PeriphRst_I2C1</code>	Управление сбросом блока I2C 1
<code>RCC_PeriphRst_USB</code>	Управление сбросом блока USB
<code>RCC_PeriphRst_Timer0</code>	Управление сбросом блока Timer 0
<code>RCC_PeriphRst_Timer1</code>	Управление сбросом блока Timer 1
<code>RCC_PeriphRst_Timer2</code>	Управление сбросом блока Timer 2
<code>RCC_PeriphRst_UART0</code>	Управление сбросом блока UART 0
<code>RCC_PeriphRst_UART1</code>	Управление сбросом блока UART 1
<code>RCC_PeriphRst_UART2</code>	Управление сбросом блока UART 2
<code>RCC_PeriphRst_UART3</code>	Управление сбросом блока UART 3
<code>RCC_PeriphRst_SPI0</code>	Управление сбросом блока SPI 0
<code>RCC_PeriphRst_SPI1</code>	Управление сбросом блока SPI 1
<code>RCC_PeriphRst_SPI2</code>	Управление сбросом блока SPI 2
<code>RCC_PeriphRst_SPI3</code>	Управление сбросом блока SPI 3
<code>RCC_PeriphRst_ETH</code>	Управление сбросом блока Ethernet
<code>RCC_PeriphRst_QEP0</code>	Управление сбросом блока QEP 0
<code>RCC_PeriphRst_QEP1</code>	Управление сбросом блока QEP 1
<code>RCC_PeriphRst_PWM0</code>	Управление сбросом блока PWM 0
<code>RCC_PeriphRst_PWM1</code>	Управление сбросом блока PWM 1
<code>RCC_PeriphRst_PWM2</code>	Управление сбросом блока PWM 2
<code>RCC_PeriphRst_PWM3</code>	Управление сбросом блока PWM 3
<code>RCC_PeriphRst_PWM4</code>	Управление сбросом блока PWM 4
<code>RCC_PeriphRst_PWM5</code>	Управление сбросом блока PWM 5
<code>RCC_PeriphRst_PWM6</code>	Управление сбросом блока PWM 6
<code>RCC_PeriphRst_PWM7</code>	Управление сбросом блока PWM 7
<code>RCC_PeriphRst_PWM8</code>	Управление сбросом блока PWM 8
<code>RCC_PeriphRst_CAP0</code>	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	<p>Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a>:</p> <ul style="list-style-type: none"> <li>• <a href="#">ENABLE</a> - переводит H[0] в выход включенной альтернативной функцией 2.</li> <li>• <a href="#">DISABLE</a> - переводит H[0] в состояние по умолчанию.</li> </ul>
-------	---

#### 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 `FIN` – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- $3,2 \text{ МГц} < FIN < 150 \text{ МГц}$ ,
- $800 \text{ КГц} < F_{REF} < 8 \text{ МГц}$ ,
- $200 \text{ МГц} < F_{VCO} < 500 \text{ МГц}$ ,

где частота фазового детектора `FREF` вычисляется по формуле:

$$F_{REF} = FIN / (2 \times NR),$$

а частота `FVCO` вычисляется по формуле:

$$F_{VCO} = FIN \times (NF / NR)$$

Аргументы

<code>RCC_PLLInit_Struct</code>	Указатель на структуру типа <a href="#">RCC_PLLInit_TypeDef</a> , 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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">RCC_PLLInit_TypeDef</a> , 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 <a href="#">RCC_PeriphClk_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

#### 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 <a href="#">RCC_SysClk_TypeDef</a> .
------------	--

## 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 <a href="#">RCC_SysClk_TypeDef</a>
----------	---------------------------------------

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 <a href="#">FunctionalState</a> .
-------	---

#### 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 <a href="#">RCC_USBFreq_TypeDef</a> .
RCC_USBFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из <a href="#">RCC_USBFreq_TypeDef</a> .

#### 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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .

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 <a href="#">RCC_UARTClk_TypeDef</a> .

## 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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .

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 <a href="#">RCC_SPIClk_TypeDef</a> .

## 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 <a href="#">RCC_ADCClk_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

#### 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 <a href="#">RCC_ADCClk_TypeDef</a> .
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 <a href="#">FunctionalState</a> .

#### 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

<a href="#">RCC_PeriphRst</a>	Выбор периферийного модуля. Параметр принимает любое значение из <a href="#">RCC_PeriphRst_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

#### 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) <= 0xFF)`  
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.

## 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 <a href="#">FunctionalState</a> .

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 <a href="#">TIMER_Ext↔ Input_TypeDef</a> .
---------------------	--

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

<a href="#">TIMERx</a>	Выбор таймера, где x лежит в диапазоне 0-2.
<a href="#">State</a>	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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

<a href="#">TIMERx</a>	Выбор таймера, где x лежит в диапазоне 0-2.
------------------------	---

Возвращаемые значения

<a href="#">Status</a>	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

<a href="#">TIMERx</a>	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_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_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_ParityBit_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_FIFOLevel_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 143

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 84

Используется в UART\_DMAMCmd() и UART\_ITFIFOLevelConfig().

#### 6.73.2.3 #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 166

Используется в UART\_Init() и UART\_ITFIFOLevelConfig().

#### 6.73.2.4 #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 121

Используется в UART\_Init().



#### 6.73.2.5 `#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 101

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 131

#### 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 74

#### 6.73.3.3 `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 153

#### 6.73.3.4 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 108

#### 6.73.3.5 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 91

## 6.74 Константы

### Группы

- [Источники прерываний UART](#)
- [Флаги работы UART](#)
- [Ошибки приемника UART](#)

#### 6.74.1 Подробное описание

## 6.75 Источники прерываний UART

### Макросы

- `#define UART_ITSource_ChangeRI ((uint32_t)0x00000001)`
- `#define UART_ITSource_ChangeCTS ((uint32_t)0x00000002)`
- `#define UART_ITSource_ChangeDCD ((uint32_t)0x00000004)`
- `#define UART_ITSource_ChangeDSR ((uint32_t)0x00000008)`
- `#define UART_ITSource_RxFIFOLevel ((uint32_t)0x00000010)`
- `#define UART_ITSource_TxFIFOLevel ((uint32_t)0x00000020)`
- `#define UART_ITSource_RecieveTimeout ((uint32_t)0x00000040)`
- `#define UART_ITSource_ErrorFrame ((uint32_t)0x00000080)`
- `#define UART_ITSource_ErrorParity ((uint32_t)0x00000100)`
- `#define UART_ITSource_ErrorBreak ((uint32_t)0x00000200)`
- `#define UART_ITSource_ErrorOverflow ((uint32_t)0x00000400)`
- `#define UART_ITSource_All ((uint32_t)0x000007FF)`
- `#define IS_UART_IT_SOURCE(IT_SOURCE) (((IT_SOURCE) & ~UART_ITSource_All) == 0)`

Макрос проверки номеров источников прерываний на попадание в допустимый диапазон.

### 6.75.1 Подробное описание

### 6.75.2 Макросы

#### 6.75.2.1 `#define UART_ITSource_All ((uint32_t)0x000007FF)`

Все источники выбраны.

См. определение в файле `niietcm4_uart.h` строка 238

#### 6.75.2.2 `#define UART_ITSource_ChangeCTS ((uint32_t)0x00000002)`

Изменение состояния линии `UART_CTS`

См. определение в файле `niietcm4_uart.h` строка 228

#### 6.75.2.3 `#define UART_ITSource_ChangeDCD ((uint32_t)0x00000004)`

Изменение состояния линии `UART_DCD`

См. определение в файле `niietcm4_uart.h` строка 229

#### 6.75.2.4 `#define UART_ITSource_ChangeDSR ((uint32_t)0x00000008)`

Изменение состояния линии `UART_DSR`

См. определение в файле `niietcm4_uart.h` строка 230

#### 6.75.2.5 `#define UART_ITSource_ChangeRI ((uint32_t)0x00000001)`

Изменение состояния линии `UART_RI`

См. определение в файле `niietcm4_uart.h` строка 227

6.75.2.6 `#define UART_ITSource_ErrorBreak ((uint32_t)0x00000200)`

Разрыв линии

См. определение в файле `niietcm4_uart.h` строка 236

6.75.2.7 `#define UART_ITSource_ErrorFrame ((uint32_t)0x00000080)`

Ошибка в структуре кадра

См. определение в файле `niietcm4_uart.h` строка 234

6.75.2.8 `#define UART_ITSource_ErrorOverflow ((uint32_t)0x00000400)`

Переполнение буфера приемника

См. определение в файле `niietcm4_uart.h` строка 237

6.75.2.9 `#define UART_ITSource_ErrorParity ((uint32_t)0x00000100)`

Ошибка контроля четности

См. определение в файле `niietcm4_uart.h` строка 235

6.75.2.10 `#define UART_ITSource_RecieveTimeout ((uint32_t)0x00000040)`

Таймаут приема данных

См. определение в файле `niietcm4_uart.h` строка 233

6.75.2.11 `#define UART_ITSource_RxFIFOLevel ((uint32_t)0x00000010)`

Порог заполнения буфера приемника

См. определение в файле `niietcm4_uart.h` строка 231

6.75.2.12 `#define UART_ITSource_TxFIFOLevel ((uint32_t)0x00000020)`

Порог опустошения буфера передатчика

См. определение в файле `niietcm4_uart.h` строка 232

## 6.76 Флаги работы UART

### Макросы

- `#define UART_Flag_InvCTS ((uint32_t)0x00000001)`
- `#define UART_Flag_InvDSR ((uint32_t)0x00000002)`
- `#define UART_Flag_InvDCD ((uint32_t)0x00000004)`
- `#define UART_Flag_Busy ((uint32_t)0x00000008)`
- `#define UART_Flag_RxFIFOEmpty ((uint32_t)0x00000010)`
- `#define UART_Flag_TxFIFOFull ((uint32_t)0x00000020)`
- `#define UART_Flag_RxFIFOFull ((uint32_t)0x00000040)`
- `#define UART_Flag_TxFIFOEmpty ((uint32_t)0x00000080)`
- `#define UART_Flag_InvRI ((uint32_t)0x00000100)`
- `#define UART_Flag_All ((uint32_t)0x000001FF)`
- `#define IS_UART_FLAG(FLAGS) (((FLAGS) & ~UART_Flag_All) == 0)`

Макрос проверки номеров флагов на попадание в допустимый диапазон.

### 6.76.1 Подробное описание

### 6.76.2 Макросы

#### 6.76.2.1 `#define UART_Flag_All ((uint32_t)0x000001FF)`

Все флаги выбраны.

См. определение в файле `niietcm4_uart.h` строка 263

#### 6.76.2.2 `#define UART_Flag_Busy ((uint32_t)0x00000008)`

Флаг занятости блока UART.

См. определение в файле `niietcm4_uart.h` строка 257

#### 6.76.2.3 `#define UART_Flag_InvCTS ((uint32_t)0x00000001)`

Флаг инверсии сигнала на линии UART\_CTS.

См. определение в файле `niietcm4_uart.h` строка 254

#### 6.76.2.4 `#define UART_Flag_InvDCD ((uint32_t)0x00000004)`

Флаг инверсии сигнала на линии UART\_DSR.

См. определение в файле `niietcm4_uart.h` строка 256

#### 6.76.2.5 `#define UART_Flag_InvDSR ((uint32_t)0x00000002)`

Флаг инверсии сигнала на линии UART\_DSR.

См. определение в файле `niietcm4_uart.h` строка 255

#### 6.76.2.6 `#define UART_Flag_InvRI ((uint32_t)0x00000100)`

Флаг инверсии сигнала на линии UART\_RI.

См. определение в файле `niietcm4_uart.h` строка 262

6.76.2.7 `#define UART_Flag_RxFIFOEmpty ((uint32_t)0x00000010)`

Флаг пустоты буфера приемника.

См. определение в файле `niietcm4_uart.h` строка 258

6.76.2.8 `#define UART_Flag_RxFIFOFull ((uint32_t)0x00000040)`

Флаг заполнения буфера приемника.

См. определение в файле `niietcm4_uart.h` строка 260

6.76.2.9 `#define UART_Flag_TxFIFOEmpty ((uint32_t)0x00000080)`

Флаг пустоты буфера передатчика.

См. определение в файле `niietcm4_uart.h` строка 261

6.76.2.10 `#define UART_Flag_TxFIFOFull ((uint32_t)0x00000020)`

Флаг заполнения буфера передатчика.

См. определение в файле `niietcm4_uart.h` строка 259

## 6.77 Ошибки приемника UART

### Макросы

- `#define UART_Error_Frame ((uint32_t)0x00000001)`
- `#define UART_Error_Parity ((uint32_t)0x00000002)`
- `#define UART_Error_Break ((uint32_t)0x00000004)`
- `#define UART_Error_Overflow ((uint32_t)0x00000008)`
- `#define UART_Error_All ((uint32_t)0x0000000F)`
- `#define IS_UART_ERROR(ERROR) (((ERROR) & ~UART_Error_All) == 0)`

Макрос проверки номеров флагов ошибок на попадание в допустимый диапазон.

#### 6.77.1 Подробное описание

#### 6.77.2 Макросы

##### 6.77.2.1 `#define UART_Error_All ((uint32_t)0x0000000F)`

Все флаги ошибок выбраны.

См. определение в файле `niietcm4_uart.h` строка 283

##### 6.77.2.2 `#define UART_Error_Break ((uint32_t)0x00000004)`

Флаг разрыва линии.

См. определение в файле `niietcm4_uart.h` строка 281

##### 6.77.2.3 `#define UART_Error_Frame ((uint32_t)0x00000001)`

Флаг ошибки в структуре кадра.

См. определение в файле `niietcm4_uart.h` строка 279

##### 6.77.2.4 `#define UART_Error_Overflow ((uint32_t)0x00000008)`

Флаг переполнения буфера приемника.

См. определение в файле `niietcm4_uart.h` строка 282

##### 6.77.2.5 `#define UART_Error_Parity ((uint32_t)0x00000002)`

Флаг ошибки контроля четности.

См. определение в файле `niietcm4_uart.h` строка 280



## 6.78 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.

#### 6.78.1 Подробное описание

#### 6.78.2 Функции

6.78.2.1 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.  
Используется в [UART\\_Init\(\)](#).

6.78.2.2 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 <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 106

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_UART\_ALL\_PERIPH.

6.78.2.3 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 69

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_UART\_ALL\_PERIPH.

## 6.79 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.

### 6.79.1 Подробное описание

### 6.79.2 Функции

#### 6.79.2.1 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 niitcm4\_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().

#### 6.79.2.2 [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
<a href="#">UART_InitStruct</a>	Указатель на структуру типа <a href="#">UART_Init_TypeDef</a> , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из <a href="#">OperationStatus</a> .
--------	---

См. определение в файле niitcm4\_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\\_BaudRateDivConfig\(\)](#), [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](#).

6.79.2.3 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 <a href="#">UART_Init_TypeDef</a> , которую необходимо проинициализировать.
----------------------	---

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_uart.c` строка 228

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.

## 6.80 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, uint32\_t UART\_Flag)  
Запрос состояния выбранного флага.
- [FlagStatus UART\\_ErrorStatus](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_Error)  
Запрос состояния выбранного флага ошибки.
- void [UART\\_ErrorStatusClear](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_Error)  
Очистка флагов ошибки.

### 6.80.1 Подробное описание

### 6.80.2 Функции

#### 6.80.2.1 [FlagStatus UART\\_ErrorStatus](#) ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Error )

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 313

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

#### 6.80.2.2 void [UART\\_ErrorStatusClear](#) ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Error )

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 339

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

6.80.2.3 FlagStatus UART\_FlagStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Flag )

Запрос состояния выбранного флага.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флагов. Параметр принимает любую совокупность значений из <a href="#">Флаги работы UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 286

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_FLAG.

6.80.2.4 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 270

Перекрестные ссылки IS\_UART\_ALL\_PERIPH.

6.80.2.5 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 249

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_DATA.



## 6.81 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.

#### 6.81.1 Подробное описание

#### 6.81.2 Функции

6.81.2.1 void [UART\\_ModemConfig](#) ( NT\_UART\_TypeDef \* UARTx, [UART\\_ModemInit\\_TypeDef](#) \* [UART\\_ModemInitStruct](#) )

Инициализирует модемный режим UART согласно параметрам структуры UART\_ModemInitStruct.

Аргументы

<a href="#">UART_ModemInitStruct</a>	Указатель на структуру типа <a href="#">UART_ModemInit_TypeDef</a> , которая содержит конфигурационную информацию.
--------------------------------------	--

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 355

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](#).

6.81.2.2 void [UART\\_ModemStructInit](#) ( [UART\\_ModemInit\\_TypeDef](#) \* [UART\\_ModemInitStruct](#) )

Заполнение каждого члена структуры UART\_ModemInitStruct значениями по умолчанию.

Аргументы

<a href="#">UART_ModemInitStruct</a>	Указатель на структуру типа <a href="#">UART_ModemInit_TypeDef</a> , которую необходимо проинициализировать.
--------------------------------------	--

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 376

Перекрестные ссылки [UART\\_ModemInit\\_TypeDef::UART\\_CTSEn](#), [UART\\_ModemInit\\_TypeDef::UART\\_InvDTR](#), [UART\\_ModemInit\\_TypeDef::UART\\_InvRTS](#) и [UART\\_ModemInit\\_TypeDef::UART\\_RTSEn](#).

## 6.82 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, uint32\_t UART\_ITSource, [FunctionalState](#) State)  
Маскирование выбранных прерываний.
- [FlagStatus](#) [UART\\_ITRawStatus](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)  
Запрос немаскированного состояния прерывания.
- [FlagStatus](#) [UART\\_ITMaskedStatus](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)  
Запрос маскированного состояния прерывания.
- void [UART\\_ITStatusClear](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)  
Сброс флагов состояния выбранных прерываний.

### 6.82.1 Подробное описание

### 6.82.2 Функции

6.82.2.1 void [UART\\_ITCmd](#) ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 <a href="#">Источники прерываний UART</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_uart.c` строка 421

Перекрестные ссылки [IS\\_UART\\_ALL\\_PERIPH](#) и [IS\\_UART\\_IT\\_SOURCE](#).

6.82.2.2 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 <a href="#">UART_Dir_Typedef</a> .

UART_FIFO↔ OLevel	Выбор порога. Параметр принимает любое значение из <a href="#">UART_FIFOLevel_↔ TypeDef</a> .
----------------------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 395

Перекрестные ссылки IS\_UART\_ALL\_PERIPH, IS\_UART\_DIR, IS\_UART\_FIFO\_LEVEL и U↔  
ART\_Dir\_Rx.

6.82.2.3 FlagStatus UART\_ITMaskedStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t  
UART\_ITSource )

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из <a href="#">Источники преры- ваний UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 472

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

6.82.2.4 FlagStatus UART\_ITRawStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t  
UART\_ITSource )

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любую совокупность значений из <a href="#">Ис- точники прерываний UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 445

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

6.82.2.5 void UART\_ITStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 любую совокупность значений из <a href="#">Ис-точники прерываний UART</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 498

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

## 6.83 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

#### 6.83.1 Подробное описание

#### 6.83.2 Функции

6.83.2.1 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 515

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_UART\_ALL\_PERIPH.

6.83.2.2 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 <a href="#">UART_Dir_Typedef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 533

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_UART\_ALL\_PERIPH, IS\_UART\_DIR и UART\_Dir\_Rx.

## 6.84 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.

#### 6.84.1 Подробное описание

## 6.85 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.

#### 6.85.1 Подробное описание

#### 6.85.2 Макросы

##### 6.85.2.1 `#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()`.

##### 6.85.2.2 `#define USERFLASH_PAGE_TOTAL ((uint32_t)256)`

Общее количество страниц.

См. определение в файле `niietcm4_userflash.h` строка 69

##### 6.85.2.3 `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`

Общий размер основной области.

См. определение в файле `niietcm4_userflash.h` строка 70

## 6.86 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.

#### 6.86.1 Подробное описание

#### 6.86.2 Макросы

##### 6.86.2.1 `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES`

Размер страницы в байтах.

См. определение в файле `niietcm4_userflash.h` строка 86

##### 6.86.2.2 `#define USERFLASH_INFO_PAGE_TOTAL ((uint32_t)2)`

Общее количество страниц.

См. определение в файле `niietcm4_userflash.h` строка 87

##### 6.86.2.3 `#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



## 6.87 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.

#### 6.87.1 Подробное описание

#### 6.87.2 Макросы

##### 6.87.2.1 `#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

#### 6.87.3 Перечисления

##### 6.87.3.1 `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

## 6.88 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.

#### 6.88.1 Подробное описание

#### 6.88.2 Функции

##### 6.88.2.1 void [USERFLASH\\_Init](#) ( uint32\_t SysClkFreq )

Инициализирует тайминги доступа для контроллера пользовательской флеш.

Аргументы

<a href="#">SysClkFreq</a>	Текущая системная частота в Гц.
----------------------------	---------------------------------

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_userflash.c строка 66

##### 6.88.2.2 void [USERFLASH\\_ITCmd](#) ( FunctionalState State )

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

<a href="#">State</a>	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-----------------------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_userflash.c строка 234

Перекрестные ссылки [IS\\_FUNCTIONAL\\_STATE](#).

##### 6.88.2.3 [USERFLASH\\_Status\\_TypeDef USERFLASH\\_OperationStatus](#) ( )

Статус работы контроллера пользовательской флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из <a href="#">USERFLASH↵ _Status_TypeDef</a> .
--------	---

См. определение в файле niietcm4\_userflash.c строка 79

Используется в USERFLASH\_FullErase(), USERFLASH\_Info\_Read() и USERFLASH\_Read().

6.88.2.4 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().

## 6.89 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.

### 6.89.1 Подробное описание

### 6.89.2 Функции

#### 6.89.2.1 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](#).

#### 6.89.2.2 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](#).

#### 6.89.2.3 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.

6.89.2.4 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.

## 6.90 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.

### 6.90.1 Подробное описание

### 6.90.2 Функции

#### 6.90.2.1 `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`.

#### 6.90.2.2 `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`.

#### 6.90.2.3 `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.

## 6.91 Типы

### Макросы

- `#define IS_WATCHDOG_RELOAD(RELOAD) ((RELOAD) > ((uint32_t)0x0))`  
Макрос проверки соответствия величины значения перезагрузки диапазону.

#### 6.91.1 Подробное описание



## 6.92 Константы

## 6.93 Функции

### Группы

- [Конфигурация](#)
- [Прерывания](#)

#### 6.93.1 Подробное описание

## 6.94 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.

#### 6.94.1 Подробное описание

#### 6.94.2 Функции

##### 6.94.2.1 void [WATCHDOG\\_Cmd](#) ( [FunctionalState](#) State )

Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_watchdog.c` строка 69

Перекрестные ссылки [IS\\_FUNCTIONAL\\_STATE](#).

##### 6.94.2.2 uint32\_t [WATCHDOG\\_GetCounter](#) ( )

Получение текущего значения счетчика.

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле `niietcm4_watchdog.c` строка 105

##### 6.94.2.3 uint32\_t [WATCHDOG\\_GetReload](#) ( )

Получение текущего значения перезагрузки.

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4\_watchdog.c строка 95

#### 6.94.2.4 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 <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_watchdog.c строка 134

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, WATCHDOG\_Lock\_Value и WATCHDOG\_Unlock\_Value.

#### 6.94.2.5 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 <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_watchdog.c строка 119

Перекрестные ссылки IS\_FUNCTIONAL\_STATE.

#### 6.94.2.6 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.

## 6.95 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.

#### 6.95.1 Подробное описание

#### 6.95.2 Функции

##### 6.95.2.1 FlagStatus WATCHDOG\_ITMaskedStatus ( )

Чтение маскированного флага прерывания сторожевого таймера.

Возвращаемые значения

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

См. определение в файле niietcm4\_watchdog.c строка 174

##### 6.95.2.2 FlagStatus WATCHDOG\_ITRawStatus ( )

Чтение немаскированного флага прерывания сторожевого таймера.

Возвращаемые значения

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

См. определение в файле niietcm4\_watchdog.c строка 153

##### 6.95.2.3 void WATCHDOG\_ITStatusClear ( )

Очищение статусного бита прерывания сторожевого таймера.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_watchdog.c строка 195

## 6.96 ADC

Драйвер для модулей АЦП, связанных с ними секвенсоров, а также цифровых компараторов.

### Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

#### 6.96.1 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.

## 6.97 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.97.1 Подробное описание

## 6.98 Приватные константы

### Группы

- [Начальные значения регистров](#)

#### 6.98.1 Подробное описание



## 6.99 Начальные значения регистров

## 6.100 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.

## 6.100.1 Подробное описание

## 6.100.2 Функции

6.100.2.1 void ADC\_Cmd ( ADC\_Module\_TypeDef ADC\_Module, FunctionalState State )

Включение модуля АЦП.

Аргументы

ADC_Module	Выбор АЦП. Параметр принимает любое значение из <a href="#">ADC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

нет
-----

См. определение в файле niietcm4\_adc.c строка 108

Перекрестные ссылки IS\_ADC\_MODULE и IS\_FUNCTIONAL\_STATE.

6.100.2.2 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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

нет
-----

См. определение в файле niietcm4\_adc.c строка 1017

Перекрестные ссылки IS\_ADC\_DC\_MODULE и IS\_FUNCTIONAL\_STATE.

6.100.2.3 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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	--

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_adc.c строка 300

Перекрестные ссылки IS\_ADC\_DC\_MODULE.

6.100.2.4 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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	---

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 1033

Перекрестные ссылки IS\_ADC\_DC\_MODULE.

```
6.100.2.5 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 <a href="#">ADC_DC_Module_TypeDef</a> .
ADC_DC_↔ InitStruct	Указатель на структуру типа <a href="#">ADC_DC_Init_TypeDef</a> , 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.

```
6.100.2.6 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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.

```
6.100.2.7 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 <a href="#">ADC_DC_Module_TypeDef</a> .
ADC_DC_↔ Mode	Режим срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Mode_TypeDef</a> .
ADC_DC_↔ Condition	Условие срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Condition_TypeDef</a> .

## 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.

```
6.100.2.8 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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().

```
6.100.2.9 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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().

```
6.100.2.10 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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 655

Перекрестные ссылки IS\_ADC\_DC\_MODULE.

6.100.2.11 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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 630

Перекрестные ссылки IS\_ADC\_DC\_MODULE.

6.100.2.12 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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

## Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 681

Перекрестные ссылки IS\_ADC\_DC\_MODULE.

6.100.2.13 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 <a href="#">ADC_DC_Init_TypeDef</a> , 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.

6.100.2.14 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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
--------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 1051

Перекрестные ссылки IS\_ADC\_DC\_MODULE.

6.100.2.15 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 <a href="#">ADC_↔ DC_Module_TypeDef</a> .
--------------------	---

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.

6.100.2.16 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 <a href="#">ADC_Module_↔ TypeDef</a> .
------------	---



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.

6.100.2.17 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 <a href="#">ADC_Module_TypeDef</a> .
ADC_InitStruct	Указатель на структуру типа <a href="#">ADC_Init_TypeDef</a> , 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.

6.100.2.18 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

нет
-----

См. определение в файле niietcm4\_adc.c строка 848

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE и IS\_FUNCTIONAL\_STATE.

6.100.2.19 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
----------------	---

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](#).

6.100.2.20 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 489

Перекрестные ссылки [IS\\_ADC\\_SEQ\\_MODULE](#) и [IS\\_FUNCTIONAL\\_STATE](#).

6.100.2.21 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
ADC_SEQ_↔ FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из <a href="#">ADC_SEQ_FIFOLevel_TypeDef</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 472

Перекрестные ссылки [IS\\_ADC\\_SEQ\\_FIFO\\_LEVEL](#) и [IS\\_ADC\\_SEQ\\_MODULE](#).

6.100.2.22 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 505

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.23 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 530

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.24 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 976

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.25 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 1001

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.26 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 936

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.27 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 961

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.28 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 898

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.29 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 880

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

```
6.100.2.30 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 916

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

```
6.100.2.31 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## Возвращаемые значения

ITCount	
---------	--

См. определение в файле niietcm4\_adc.c строка 750

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

```
6.100.2.32 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа <a href="#">ADC_SEQ_Init_TypeDef</a> , 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](#).

6.100.2.33 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

нет
-----

См. определение в файле niietcm4\_adc.c строка 697

Перекрестные ссылки [IS\\_ADC\\_SEQ\\_MODULE](#) и [IS\\_FUNCTIONAL\\_STATE](#).

6.100.2.34 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
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 <a href="#">ADC_SEQ_ITCountRst</a> . Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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](#).

6.100.2.35 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## Возвращаемые значения

нет
-----

См. определение в файле niietcm4\_adc.c строка 768

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.36 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 807

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.37 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## Возвращаемые значения

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

См. определение в файле niietcm4\_adc.c строка 782

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.38 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 832

Перекрестные ссылки IS\_ADC\_SEQ\_MODULE.

6.100.2.39 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 <a href="#">ADC_SEQ_Init_TypeDef</a> , 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.

6.100.2.40 void ADC\_SEQ\_SWReq ( )

Программный запуск измерений всех разрешенных секвенсоров.

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4\_adc.c строка 868

6.100.2.41 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 <a href="#">ADC_Init_TypeDef</a> , 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.



## 6.101 BOOTFLASH

Драйвер для загрузочной флеш-памяти.

### Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.101.1 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.

## 6.102 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

#### 6.102.1 Подробное описание

## 6.103 Приватные константы

## 6.104 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.

### 6.104.1 Подробное описание

### 6.104.2 Функции

#### 6.104.2.1 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](#).

#### 6.104.2.2 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.

6.104.2.3 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.

6.104.2.4 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

6.104.2.5 void BOOTFLASH\_ITCmd ( FunctionalState State )

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_bootflash.c строка 194

Перекрестные ссылки IS\_FUNCTIONAL\_STATE.

6.104.2.6 BOOTFLASH\_Status\_TypeDef BOOTFLASH\_OperationStatus ( )

Статус работы контроллера загрузочной флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из <a href="#">BOOTFLASH_H_Status_TypeDef</a> .
--------	---

См. определение в файле niietcm4\_bootflash.c строка 88

6.104.2.7 void BOOTFLASH\_OperationStatusClear ( )

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4\_bootflash.c строка 102

6.104.2.8 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.

6.104.2.9 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.

## 6.105 CAP

Драйвер для блоков захвата

### Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.105.1 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](#).

## 6.106 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.106.1 Подробное описание



## 6.107 Приватные константы

## 6.108 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.

### 6.108.1 Подробное описание

### 6.108.2 Функции

#### 6.108.2.1 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_cap.c строка 444

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_FUNCTIONAL\_STATE.

#### 6.108.2.2 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.

6.108.2.3 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.

6.108.2.4 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.

6.108.2.5 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.

6.108.2.6 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 <a href="#">CAP_Capture_Init_TypeDef</a> , 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`.

6.108.2.7 `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`.

6.108.2.8 `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`.

6.108.2.9 `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`.

6.108.2.10 `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`.

6.108.2.11 `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`.

6.108.2.12 `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().

6.108.2.13 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.

6.108.2.14 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.

6.108.2.15 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.



6.108.2.16 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 <a href="#">Маски источников прерываний</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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.

6.108.2.17 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 <a href="#">Маски источников прерываний</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_cap.c строка 599

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_CAP\_IT\_SOURCE.

6.108.2.18 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 681

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH.

6.108.2.19 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 657

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH.

6.108.2.20 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 <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Status	Статус прерывания. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_cap.c строка 616

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_CAP\_IT\_SOURCE.

6.108.2.21 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 <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_cap.c строка 643

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_CAP\_IT\_SOURCE.

6.108.2.22 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.

6.108.2.23 `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`.

6.108.2.24 `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`.

6.108.2.25 `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`.

6.108.2.26 `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 <a href="#">CAP_PWM_Init_TypeDef</a> , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет	
-----	--

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

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

6.108.2.27 `void CAP_PWM_SetCompare ( NT_CAP_TypeDef * CAPx, uint32_t CompareVal )`

Установка значения сравнения ШИМ.

Аргументы

<code>CAPx</code>	Выбор таймера, где x лежит в диапазоне 0-5.
<code>CompareVal</code>	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()`.

6.108.2.28 `void CAP_PWM_SetPeriod ( NT_CAP_TypeDef * CAPx, uint32_t PeriodVal )`

Установка значения периода ШИМ.

Аргументы

<code>CAPx</code>	Выбор таймера, где x лежит в диапазоне 0-5.
<code>PeriodVal</code>	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()`.

6.108.2.29 `void CAP_PWM_SetShadowCompare ( NT_CAP_TypeDef * CAPx, uint32_t CompareVal )`

Установка значения сравнения ШИМ для отложенной записи.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_cap.c строка 319

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH.

6.108.2.30 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.

6.108.2.31 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.

6.108.2.32 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.

6.108.2.33 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.

6.108.2.34 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\_P\_Mode, CAP\_Mode\_Capture, CAP\_Init\_TypeDef::CAP\_SyncCmd, CAP\_Init\_TypeDef::CAP\_SyncOut и CAP\_SyncOut\_Bypass.

6.108.2.35 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.

6.108.2.36 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.

6.108.2.37 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_cap.c строка 163

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_FUNCTIONAL\_STATE.

## 6.109 DMA

Драйвер для работы с контроллером прямого доступа памяти.

### Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.109.1 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.



## 6.110 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.110.1 Подробное описание

## 6.111 Приватные константы

### Группы

- [Начальные значения регистров](#)

#### 6.111.1 Подробное описание

## 6.112 Начальные значения регистров

## 6.113 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.

### 6.113.1 Подробное описание

### 6.113.2 Функции

#### 6.113.2.1 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

6.113.2.2 `void DMA_ChannelDeInit ( DMA_Channel_TypeDef * DMA_Channel )`

Деинициализация канала DMA.

## Аргументы

DMA_Channel	Указатель на структуру типа <a href="#">DMA_Channel_TypeDef</a> , 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`.

6.113.2.3 `void DMA_ChannelEnableCmd ( uint32_t DMA_Channel, FunctionalState State )`

Активация каналов DMA.

## Аргументы

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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()`.

6.113.2.4 `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 <a href="#">DMA_ChannelInitStruct_TypeDef</a> , 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.

6.113.2.5 void DMA\_ChannelStructInit ( DMA\_ChannelInit\_TypeDef \* DMA\_ChannelInitStruct )

Заполнение каждого члена структуры DMA\_ChannelInitStruct значениями по умолчанию.

Аргументы

DMA_ChannelInitStruct	Указатель на структуру типа <a href="#">DMA_ChannelInit_TypeDef</a> , 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.

6.113.2.6 void DMA\_ClearErrorStatus ( )

Сброс флага ошибки на шине.

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_dma.c строка 524

#### 6.113.2.7 void DMA\_DeInit ( )

Деинициализация контроллера DMA.

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_dma.c строка 174

#### 6.113.2.8 OperationStatus DMA\_ErrorStatus ( )

Показывает наличие ошибки на шине.

Возвращаемые значения

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

См. определение в файле niietcm4\_dma.c строка 503

#### 6.113.2.9 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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().

#### 6.113.2.10 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 <a href="#">DMA_Init_TypeDef</a> , 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\(\)](#).

6.113.2.11 `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 <a href="#">FunctionalState</a> .
-------	---

## Возвращаемые значения

Нет
-----

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

Перекрестные ссылки [IS\\_FUNCTIONAL\\_STATE](#).

6.113.2.12 `FunctionalState DMA_MasterEnableStatus ( )`

Состояние контроллера DMA.

## Возвращаемые значения

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

См. определение в файле `niietcm4_dma.c` строка 455

6.113.2.13 `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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .



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()`.

6.113.2.14 `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 <a href="#">DMA_Protect_TypeDef</a> .
-----------------------------------	--

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()`.

6.113.2.15 `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 <a href="#">Маски каналов по номеру</a> .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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()`.

6.113.2.16 `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

6.113.2.17 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 <a href="#">DMA_Init_TypeDef</a> , 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`.

6.113.2.18 `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 <a href="#">Маски каналов по номеру</a> .
-------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_dma.c` строка 308

Перекрестные ссылки `IS_DMA_CHANNEL`.

6.113.2.19 `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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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()`.

6.113.2.20 `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"><li>• ENABLE - поддерживаются SREQ (как и блочные BREQ);</li><li>• DISABLE - поддерживаются только блочные запросы BREQ.</li></ul>
--------	--

См. определение в файле niietcm4\_dma.c строка 478

Перекрестные ссылки IS\_GET\_DMA\_CHANNEL.

## 6.114 EXTMEM

Драйвер для интерфейса внешней памяти.

### Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.114.1 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.

## 6.115 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

#### 6.115.1 Подробное описание

## 6.116 Приватные константы

### Группы

- [Начальные значения регистров](#)

#### 6.116.1 Подробное описание

## 6.117 Начальные значения регистров

### Макросы

- `#define EXT_MEM_CFG_Reset_Value ((uint32_t)0x80000007)`

#### 6.117.1 Подробное описание

#### 6.117.2 Макросы

##### 6.117.2.1 `#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()`.



## 6.118 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.

### 6.118.1 Подробное описание

### 6.118.2 Функции

#### 6.118.2.1 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](#).

#### 6.118.2.2 void [EXTMEM\\_Init](#) ( [EXTMEM\\_Init\\_TypeDef](#) \* [EXTMEM\\_InitStruct](#) )

Инициализирует внешнюю память согласно параметрам структуры [EXTMEM\\_InitStruct](#).

Аргументы

<a href="#">EXTMEM_InitStruct</a>	Указатель на структуру типа <a href="#">EXTMEM_Init_TypeDef</a> , 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](#).

#### 6.118.2.3 void [EXTMEM\\_StructInit](#) ( [EXTMEM\\_Init\\_TypeDef](#) \* [EXTMEM\\_InitStruct](#) )

Заполнение каждого члена структуры [EXTMEM\\_InitStruct](#) значениями по умолчанию.

Аргументы

<a href="#">EXTMEM_InitStruct</a>	Указатель на структуру типа <a href="#">EXTMEM_Init_TypeDef</a> , 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.

## 6.119 GPIO

Драйвер для управления портами ввода-вывода.

### Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

#### 6.119.1 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.

## 6.120 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.120.1 Подробное описание

## 6.121 Приватные константы

### Группы

- [Начальные значения регистров](#)
- [Маски портов](#)

### 6.121.1 Подробное описание

## 6.122 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)`

### 6.122.1 Подробное описание

### 6.122.2 Макросы

6.122.2.1 `#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()`.

6.122.2.2 `#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()`.

6.122.2.3 `#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()`.

6.122.2.4 `#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()`.

6.122.2.5 `#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().

6.122.2.6 `#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().

6.122.2.7 `#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().

6.122.2.8 `#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().

6.122.2.9 `#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().

6.122.2.10 `#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().

6.122.2.11 `#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().

6.122.2.12 `#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().

6.122.2.13 `#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()`.



## 6.123 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)`

#### 6.123.1 Подробное описание

#### 6.123.2 Макросы

##### 6.123.2.1 `#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()`.

##### 6.123.2.2 `#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()`.

##### 6.123.2.3 `#define GPIO_Regs_GPIOA_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOA

См. определение в файле `niietcm4_gpio.c` строка 96

##### 6.123.2.4 `#define GPIO_Regs_GPIOB_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOB

См. определение в файле `niietcm4_gpio.c` строка 97

##### 6.123.2.5 `#define GPIO_Regs_GPIOC_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOC

См. определение в файле `niietcm4_gpio.c` строка 98

##### 6.123.2.6 `#define GPIO_Regs_GPIOD_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOD

См. определение в файле `niietcm4_gpio.c` строка 99

6.123.2.7 `#define GPIO_Regs_GPIOE_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOE

См. определение в файле `niietcm4_gpio.c` строка 100

6.123.2.8 `#define GPIO_Regs_GPIOF_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOF

См. определение в файле `niietcm4_gpio.c` строка 101

6.123.2.9 `#define GPIO_Regs_GPIOG_Mask ((uint32_t)0x0000FFFF)`

Маска регистров для порта GPIOG

См. определение в файле `niietcm4_gpio.c` строка 102

6.123.2.10 `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOH

См. определение в файле `niietcm4_gpio.c` строка 103

## 6.124 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)  
Осуществляет выбор альтернативной функции вывода GPIOx.
- 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 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.

### 6.124.1 Подробное описание

### 6.124.2 Функции

6.124.2.1 `void GPIO_AltFuncConfig ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,  
GPIO_AltFunc_TypeDef GPIO_AltFunc )`

Осуществляет выбор альтернативной функции вывода GPIOx.

## 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.
GPIO_AltFunc	Выбор номера альтернативной функции пина. Параметр принимает любое значение из <a href="#">GPIO_AltFunc_TypeDef</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_gpio.c` строка 278

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`, `IS_GPIO_ALT_FUNC` и `IS_GPIO_PIN`.

Используется в `GPIO_Init()`.

6.124.2.2 `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 635

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.124.2.3 `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`.

6.124.2.4 `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_Init↔ Struct	Указатель на структуру типа <a href="#">GPIO_Init_TypeDef</a> , которая содержит конфигурационную информацию.

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_gpio.c` строка 340

Перекрестные ссылки `GPIO_Init_TypeDef::GPIO_AltFunc`, `GPIO_AltFuncConfig()`, `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`, `GP↔IO_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::GP↔IO_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()`.

```
6.124.2.5 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 <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_gpio.c` строка 922

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

```
6.124.2.6 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 <a href="#">GPIO_IntType_TypeDef</a> .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из <a href="#">GPIO_IntPol_TypeDef</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 885

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.

6.124.2.7 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 947

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

6.124.2.8 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 762

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

6.124.2.9 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 <a href="#">GPIO_QualMode_TypeDef</a> .
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 673

Перекрестные ссылки `GPIO_QualMode_3sample`, `GPIO_QualMode_6sample`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_PIN`, `IS_GPIO_QUAL_MODE` и `IS_GPIO_QUAL_PERIOD`.

6.124.2.10 `uint32_t GPIO_Read ( NT_GPIO_TypeDef * GPIOx )`

Чтение состояния выбранного порта `GPIOx`.

Аргументы

<code>GPIOx</code>	Выбор порта, где <code>x</code> лежит в диапазоне А..Н.
--------------------	---

Возвращаемые значения

Состояние порта <code>GPIOx</code>
------------------------------------

См. определение в файле `niietcm4_gpio.c` строка 512

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

6.124.2.11 `uint32_t GPIO_ReadBit ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin )`

Чтение состояния выбранного пина.

Аргументы

<code>GPIOx</code>	Выбор порта, где <code>x</code> лежит в диапазоне А..Н.
<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 0..15.

Возвращаемые значения

Состояние выбранного пина
---------------------------

См. определение в файле `niietcm4_gpio.c` строка 486

Перекрестные ссылки `Bit_CLEAR`, `Bit_SET`, `IS_GET_GPIO_PIN` и `IS_GPIO_ALL_PERIPH`.

6.124.2.12 `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 А..Н.
<code>MaskVal</code>	Значение маски чтения.



Возвращаемые значения

Состояние порта GPIOx с учетом маски	
---	--

См. определение в файле niietcm4\_gpio.c строка 527

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

6.124.2.13 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 618

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

6.124.2.14 void GPIO\_StructInit ( GPIO\_Init\_TypeDef \* GPIO\_InitStruct )

Заполнение каждого члена структуры GPIO\_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа <a href="#">GPIO_Init_TypeDef</a> , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 465

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().

6.124.2.15 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 <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 823

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

6.124.2.16 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 652

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

6.124.2.17 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 579

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

6.124.2.18 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 <a href="#">BitAction</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 555

Перекрестные ссылки `Bit_CLEAR`, `Bit_SET`, `IS_GET_GPIO_PIN`, `IS_GPIO_ALL_PERIPH` и `IS_GPIO_BIT_ACTION`.

6.124.2.19 `void GPIO_WriteMask ( NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal )`

Изменение состояния выбранного порта `GPIOx` с использованием маски.

Аргументы

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне A..H.
<code>MaskVal</code>	Значение маски.
<code>PortVal</code>	Значение которое будет записано.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_gpio.c` строка 595

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

## 6.125 RCC

Драйвер для работы с тактированием и сбросом периферийных блоков.

### Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.125.1 Подробное описание

Драйвер для работы с тактированием и сбросом периферийных блоков.

## 6.126 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.126.1 Подробное описание

## 6.127 Приватные константы

### Группы

- [Начальные значения регистров](#)

#### 6.127.1 Подробное описание

## 6.128 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)`

#### 6.128.1 Подробное описание

#### 6.128.2 Макросы

##### 6.128.2.1 `#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()`.

##### 6.128.2.2 `#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()`.

##### 6.128.2.3 `#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()`.

##### 6.128.2.4 `#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()`.

## 6.129 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.



## 6.129.1 Подробное описание

## 6.129.2 Функции

6.129.2.1 void RCC\_ADCClkCmd ( RCC\_ADCClk\_TypeDef RCC\_ADCClk, FunctionalState State )

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из <a href="#">RCC_ADCClk_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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.

6.129.2.2 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 <a href="#">RCC_ADCClk_TypeDef</a> .
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 <a href="#">FunctionalState</a> .

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.

6.129.2.3 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 <a href="#">RCC_PeriphClk_↔TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_rcc.c` строка 342

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_RCC_PERIPH_CLK`.

6.129.2.4 `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 <a href="#">RCC_↔_PeriphRst_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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()`.

6.129.2.5 `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  $\text{Freq} < 50 \text{ МГц}$ , 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 <a href="#">RCC_PLLRef_TypeDef</a> .
------------	---

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().

#### 6.129.2.6 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.

#### 6.129.2.7 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 <a href="#">RCC_PLLInit_TypeDef</a> , 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().

6.129.2.8 void RCC\_PLLPowerDownCmd ( FunctionalState State )

Управление режимом PowerDown PLL.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_rcc.c строка 313

Перекрестные ссылки IS\_FUNCTIONAL\_STATE.

6.129.2.9 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 <a href="#">RCC_PLLInit_TypeDef</a> , 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.

6.129.2.10 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 <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_rcc.c` строка 648

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_SPI_ALL_PERIPH`.

6.129.2.11 `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 <a href="#">FunctionalState</a> .

## 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`.

6.129.2.12 `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 <a href="#">RCC_SPIClk_TypeDef</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_rcc.c` строка 569

Перекрестные ссылки `IS_RCC_SPI_CLK` и `IS_SPI_ALL_PERIPH`.

6.129.2.13 `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"> <li>• ENABLE - переводит H[0] в выход включенной альтернативной функцией 2.</li> <li>• DISABLE - переводит H[0] в состояние по умолчанию.</li> </ul>
-------	--

## 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.

6.129.2.14 OperationStatus RCC\_SysClkSel ( RCC\_SysClk\_TypeDef RCC\_SysClk )

Выбор источника для системного тактового сигнала.

## Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из <a href="#">RCC_SysClk_TypeDef</a> .
------------	--

## 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().

6.129.2.15 RCC\_SysClk\_TypeDef RCC\_SysClkStatus ( )

Текущий источник системного тактового сигнала.

## Возвращаемые значения

Значение	из <a href="#">RCC_SysClk_TypeDef</a>
----------	---------------------------------------

См. определение в файле niietcm4\_rcc.c строка 394

6.129.2.16 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 526

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.129.2.17 `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 <a href="#">FunctionalState</a> .

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`.

6.129.2.18 `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 <a href="#">RCC_UARTClk_TypeDef</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 448

Перекрестные ссылки `IS_RCC_UART_CLK` и `IS_UART_ALL_PERIPH`.

6.129.2.19 `void RCC_USBClkCmd ( FunctionalState State )`

Включение тактирования USB.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_rcc.c строка 425

Перекрестные ссылки IS\_FUNCTIONAL\_STATE.

```
6.129.2.20 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 <a href="#">RCC_C_USBClk_TypeDef</a> .
RCC_USB↔Freq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из <a href="#">RCC_USBFreq_TypeDef</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_rcc.c строка 408

Перекрестные ссылки IS\_RCC\_USB\_CLK и IS\_RCC\_USB\_FREQ.

```
6.129.2.21 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().



## 6.130 RTC

Драйвер для модуля часов реального времени.

### Группы

- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.130.1 Подробное описание

Драйвер для модуля часов реального времени.

## 6.131 Приватные данные

### Группы

- [Приватные функции](#)

#### 6.131.1 Подробное описание

## 6.132 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)

### 6.132.1 Подробное описание

## 6.133 TIMER

Драйвер для таймеров

Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.133.1 Подробное описание

Драйвер для таймеров

## 6.134 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.134.1 Подробное описание

## 6.135 Приватные константы

## 6.136 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.

### 6.136.1 Подробное описание

### 6.136.2 Функции

#### 6.136.2.1 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_timer.c строка 65

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_TIMER\_ALL\_PERIPH.

#### 6.136.2.2 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 <a href="#">TIMER_Ext↔ Input_TypeDef</a> .

## 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.

6.136.2.3 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.

6.136.2.4 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.

6.136.2.5 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.

6.136.2.6 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_timer.c строка 200

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_TIMER\_ALL\_PERIPH.

6.136.2.7 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.

6.136.2.8 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.

6.136.2.9 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.

6.136.2.10 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.

6.136.2.11 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.

## 6.137 UART

Драйвер для приемопередатчиков UART.

### Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.137.1 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](#).

## 6.138 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.138.1 Подробное описание

## 6.139 Приватные константы

## 6.140 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 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, uint32\_t UART\_Flag)  
Запрос состояния выбранного флага.
- **FlagStatus** **UART\_ErrorStatus** (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_Error)  
Запрос состояния выбранного флага ошибки.
- void **UART\_ErrorStatusClear** (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_Error)  
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, uint32\_t UART\_ITSource, **FunctionalState** State)  
Маскирование выбранных прерываний.
- **FlagStatus** **UART\_ITRawStatus** (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)  
Запрос немаскированного состояния прерывания.
- **FlagStatus** **UART\_ITMaskedStatus** (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)  
Запрос маскированного состояния прерывания.
- void **UART\_ITStatusClear** (NT\_UART\_TypeDef \*UARTx, uint32\_t 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

## 6.140.1 Подробное описание

## 6.140.2 Функции

6.140.2.1 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.

Используется в UART\_Init().

6.140.2.2 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 106

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_UART\_ALL\_PERIPH.

6.140.2.3 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 69

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_UART\_ALL\_PERIPH.

6.140.2.4 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().

6.140.2.5 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 <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 515

Перекрестные ссылки IS\_FUNCTIONAL\_STATE и IS\_UART\_ALL\_PERIPH.

6.140.2.6 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 <a href="#">UART_Dir_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 533

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_UART\_ALL\_PERIPH, IS\_UART\_DIR и UART\_Dir\_Rx.

6.140.2.7 FlagStatus UART\_ErrorStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Error )

Запрос состояния выбранного флага ошибки.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 313

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

6.140.2.8 void UART\_ErrorStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Error )

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 339

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

6.140.2.9 FlagStatus UART\_FlagStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Flag )

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флагов. Параметр принимает любую совокупность значений из <a href="#">Флаги работы UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 286

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_FLAG.

6.140.2.10 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 <a href="#">UART_Init_TypeDef</a> , которая содержит конфигурационную информацию.
----------------------	---

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из <a href="#">OperationStatus</a> .
--------	---

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\_BaudRateDivConfig(), UART\_Init\_TypeDef::UART\_↔ClkFreq, UART\_Init\_TypeDef::UART\_DataWidth, UART\_Init\_TypeDef::UART\_FIFOEn, UAR↔T\_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.

6.140.2.11 void UART\_ITCmd ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 <a href="#">Источники прерываний UART</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 421

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

6.140.2.12 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 <a href="#">UART_Dir_TypeDef</a> .
UART_FIF↔ OLevel	Выбор порога. Параметр принимает любое значение из <a href="#">UART_FIFOLevel_↔_TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 395

Перекрестные ссылки IS\_UART\_ALL\_PERIPH, IS\_UART\_DIR, IS\_UART\_FIFO\_LEVEL и U↔ART\_Dir\_Rx.

6.140.2.13    FlagStatus UART\_ITMaskedStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t  
                  UART\_ITSource )

Запрос маскированного состояния прерывания.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 472

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

6.140.2.14 FlagStatus UART\_ITRawStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_ITSource )

Запрос немаскированного состояния прерывания.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любую совокупность значений из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 445

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

6.140.2.15 void UART\_ITStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 любую совокупность значений из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 498

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

6.140.2.16 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 <a href="#">UART_ModemInit_TypeDef</a> , которая содержит конфигурационную информацию.
---------------------------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 355

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.

6.140.2.17 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 <a href="#">UART_ModemInit_TypeDef</a> , которую необходимо проинициализировать.
---------------------------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 376

Перекрестные ссылки UART\_ModemInit\_TypeDef::UART\_CTSEn, UART\_ModemInit\_TypeDef↔::UART\_InvDTR, UART\_ModemInit\_TypeDef::UART\_InvRTS и UART\_ModemInit\_TypeDef↔::UART\_RTSEn.

6.140.2.18 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 270

Перекрестные ссылки IS\_UART\_ALL\_PERIPH.

6.140.2.19 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 249

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_DATA.

6.140.2.20 void UART\_StructInit ( UART\_Init\_TypeDef \* UART\_InitStruct )

Заполнение каждого члена структуры UART\_InitStruct значениями по умолчанию.

Аргументы

UART_InitStruct	Указатель на структуру типа <a href="#">UART_Init_TypeDef</a> , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 228

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.

## 6.141 USERFLASH

Драйвер для пользовательской флеш-памяти.

### Группы

- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.141.1 Подробное описание

Драйвер для пользовательской флеш-памяти.

#### Внимание

Перед использованием какой либо функции этого драйвера, следует обязательно произвести инициализацию контроллера памяти - [USERFLASH\\_Init\(\)](#).



## 6.142 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

### 6.142.1 Подробное описание

## 6.143 Приватные константы

## 6.144 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.

### 6.144.1 Подробное описание

### 6.144.2 Функции

#### 6.144.2.1 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**.

#### 6.144.2.2 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.

6.144.2.3 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.

6.144.2.4 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.

6.144.2.5 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

6.144.2.6 void USERFLASH\_ITCmd ( FunctionalState State )

Включение прерывания по завершению чтения/записи/стирания.

## Аргументы

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_userflash.c строка 234

Перекрестные ссылки IS\_FUNCTIONAL\_STATE.

## 6.144.2.7 USERFLASH\_Status\_TypeDef USERFLASH\_OperationStatus ( )

Статус работы контроллера пользовательской флэш.

## Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из <a href="#">USERFLASH_Status_TypeDef</a> .
--------	---

См. определение в файле niietcm4\_userflash.c строка 79

Используется в USERFLASH\_FullErase(), USERFLASH\_Info\_Read() и USERFLASH\_Read().

## 6.144.2.8 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().

## 6.144.2.9 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.

## 6.144.2.10 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.

6.144.2.11 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.

## 6.145 WATCHDOG

Драйвер для сторожевого таймера

### Группы

- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

### 6.145.1 Подробное описание

Драйвер для сторожевого таймера

## 6.146 Приватные данные

### Группы

- [Приватные константы](#)
- [Приватные функции](#)

#### 6.146.1 Подробное описание



## 6.147 Приватные константы

### Макросы

- `#define WATCHDOG_Lock_Value ((uint32_t)0xDEADC0DE)`
- `#define WATCHDOG_Unlock_Value ((uint32_t)0x1ACCE551)`

#### 6.147.1 Подробное описание

#### 6.147.2 Макросы

##### 6.147.2.1 `#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()`.

##### 6.147.2.2 `#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()`.

## 6.148 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.

### 6.148.1 Подробное описание

### 6.148.2 Функции

#### 6.148.2.1 void [WATCHDOG\\_Cmd](#) ( [FunctionalState](#) State )

Разрешение счета сторожевого таймера и маскирование (включение) его прерывания.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_watchdog.c строка 69

Перекрестные ссылки [IS\\_FUNCTIONAL\\_STATE](#).

#### 6.148.2.2 uint32\_t [WATCHDOG\\_GetCounter](#) ( )

Получение текущего значения счетчика.

Возвращаемые значения

CounterVal	Значение счетчика.
------------	--------------------

См. определение в файле niietcm4\_watchdog.c строка 105

#### 6.148.2.3 uint32\_t [WATCHDOG\\_GetReload](#) ( )

Получение текущего значения перезагрузки.

Возвращаемые значения

ReloadVal	Значение перезагрузки.
-----------	------------------------

См. определение в файле niietcm4\_watchdog.c строка 95

#### 6.148.2.4 FlagStatus WATCHDOG\_ITMaskedStatus ( )

Чтение маскированного флага прерывания сторожевого таймера.

Возвращаемые значения

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

См. определение в файле niietcm4\_watchdog.c строка 174

#### 6.148.2.5 FlagStatus WATCHDOG\_ITRawStatus ( )

Чтение немаскированного флага прерывания сторожевого таймера.

Возвращаемые значения

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

См. определение в файле niietcm4\_watchdog.c строка 153

#### 6.148.2.6 void WATCHDOG\_ITStatusClear ( )

Очищение статусного бита прерывания сторожевого таймера.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_watchdog.c строка 195

#### 6.148.2.7 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 <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_watchdog.c строка 134

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, WATCHDOG\_Lock\_Value и WATCHDOG\_Unlock\_Value.

#### 6.148.2.8 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 <a href="#">FunctionalState</a> .
-------	---

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_watchdog.c строка 119

Перекрестные ссылки IS\_FUNCTIONAL\_STATE.

6.148.2.9 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.

## 6.149 Драйвер периферии

### Группы

- [Настройка драйвера](#)
- [Макросы](#)
- [Типы](#)
- [Периферия](#)

### 6.149.1 Подробное описание

## 6.150 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.

### 6.150.1 Подробное описание

### 6.150.2 Макросы

#### 6.150.2.1 `#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()`.

#### 6.150.2.2 `#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

## 6.151 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.

#### 6.151.1 Подробное описание

## 6.152 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.

#### 6.152.1 Подробное описание

#### 6.152.2 Макросы

##### 6.152.2.1 `#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()`.



## 6.152.2.2 #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\_AltFuncConfig(), 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().

## 6.152.2.3 #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().

## 6.152.2.4 #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().

## 6.152.2.5 #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_DMAMCmd()`, `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()`.

## 6.153 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

### 6.153.1 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: 0xXXXXXX000, 0xXXXXXX400, 0xXXXXXX800, 0xXXXXXXC00.

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 192

## 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 201

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 200

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 198

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 207

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 203

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 205

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 196

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 209

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 194

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 211

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 176

### 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 184

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 178

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 180

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 182

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@niet.ru](mailto:kolbov@niet.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\_AltFuncConfig(), 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](mailto: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 <a href="#">ADC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	---



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 <a href="#">ADC_DC_Module_TypeDef</a> .
ADC_DC_↔ InitStruct	Указатель на структуру типа <a href="#">ADC_DC_Init_TypeDef</a> , 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">ADC_DC_Module_TypeDef</a> .
ADC_DC_↔ Mode	Режим срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Mode_TypeDef</a> .
ADC_DC_↔ Condition	Условие срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Condition_TypeDef</a> .

## 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

## 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 <a href="#">ADC_DC_Init_TypeDef</a> , 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 <a href="#">ADC_DC_↔Module_TypeDef</a> .
--------------------	--

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 <a href="#">ADC_↔C_DC_Module_TypeDef</a> .
--------------------	--

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 <a href="#">ADC_Module_↔TypeDef</a> .
------------	--

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 <a href="#">ADC_Module_TypeDef</a> .
ADC_InitStruct	Указатель на структуру типа <a href="#">ADC_Init_TypeDef</a> , 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
----------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
ADC_SEQ_↔ FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из <a href="#">ADC_SEQ_FIFOLevel_TypeDef</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---



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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа <a href="#">ADC_SEQ_Init_TypeDef</a> , 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
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 <a href="#">ADC_SEQ_IT↔ CountRst</a> . Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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.

8.2.2.35 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_Init_TypeDef</a> , 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 <a href="#">ADC_Init_TypeDef</a> , 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)0x000000))

```

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](mailto: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) || \
((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()`.

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 <a href="#">ADC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">ADC_DC_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	--

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 <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	---

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 <a href="#">ADC_DC_Module_TypeDef</a> .
<code>ADC_DC_↵ InitStruct</code>	Указатель на структуру типа <a href="#">ADC_DC_Init_TypeDef</a> , 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 <a href="#">ADC_DC_Module_TypeDef</a> .
<code>State</code>	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">ADC_DC_Module_TypeDef</a> .
<code>ADC_DC_↵ Mode</code>	Режим срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Mode_TypeDef</a> .
<code>ADC_DC_↵ Condition</code>	Условие срабатывания компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Condition_TypeDef</a> .

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

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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

ADC_DC_↔ Module	Выбор цифрового компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_Module_TypeDef</a> .
--------------------	---

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 <a href="#">ADC_↔ C_DC_Module_TypeDef</a> .
--------------------	---

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 <a href="#">ADC_↔ C_DC_Module_TypeDef</a> .
--------------------	---

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 <a href="#">ADC_DC_Init_TypeDef</a> , 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\_TrigStatus ( ADC\_DC\_Module\_TypeDef ADC\_DC\_Module )

Проверка состояния выходного триггера компаратора.



## Аргументы

ADC_DC_↔ Module	Выбор компаратора. Параметр принимает любое значение из <a href="#">ADC_DC_↔ Module_TypeDef</a> .
--------------------	---

## 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 <a href="#">ADC_↔ C_DC_Module_TypeDef</a> .
--------------------	---

## 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 <a href="#">ADC_Module_↔ TypeDef</a> .
------------	---

## 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 <a href="#">ADC_Module_TypeDef</a> .
ADC_InitStruct	Указатель на структуру типа <a href="#">ADC_Init_TypeDef</a> , 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_Module_TypeDef</a> .
----------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
ADC_SEQ_↔ FIFOLevel	Количество результатов измерений записанных в буфер секвенсора, по достижению которого вызывается DMA. Параметр принимает любое значение из <a href="#">ADC_SEQ_FIFOLevel_TypeDef</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа <a href="#">ADC_SEQ_Init_TypeDef</a> , 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
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 <a href="#">ADC_SEQ_IT↔ CountRst</a> . Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_↔ Module_TypeDef</a> .
---------------------	---

## 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 <a href="#">ADC_SEQ_Init_TypeDef</a> , 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](mailto: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 <a href="#">FunctionalState</a> .
-------	---

## 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 <a href="#">BOOTFLASH_Status_TypeDef</a> .
--------	---

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@niiet.ru](mailto: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"

### 8.5.2 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 <a href="#">FunctionalState</a> .
-------	---

## 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 <a href="#">BOOTFLASH_Status_TypeDef</a> .
--------	---

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\_↵  
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.

### 8.6.1 Подробное описание

Файл содержит реализацию всех функции для работы с блоками захвата

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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 <a href="#">FunctionalState</a> .

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_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`.

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 <a href="#">CAP_Init_TypeDef</a> , 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 <a href="#">Маски источников прерываний</a> .

State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">Маски источников прерываний</a> .

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 681

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 657

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 <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Status	Статус прерывания. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_cap.c строка 616

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_CAP\_IT\_SOURCE.

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 <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_cap.c строка 643

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 <a href="#">CAP_PWM_Init_TypeDef</a> , 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 <a href="#">CAP_Init_TypeDef</a> , 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 <a href="#">FunctionalState</a> .

## 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 <a href="#">FunctionalState</a> .

## 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\(IT\\_SOURCE\) \(\(\(IT\\_SOURCE\) & ~CAP\\_ITSource\\_All\) == 0\)](#)  
Макрос проверки источников прерываний 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.

## 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](mailto: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 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.

© 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_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()`.

8.7.2.14 `#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()`.

8.7.2.15 `#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 <a href="#">FunctionalState</a> .

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.
------	---



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 <a href="#">CAP_Init_TypeDef</a> , 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_ITSource	Выбор источников прерывания. Параметр принимает любую совокупность значений из <a href="#">Маски источников прерываний</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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_ITSource	Выбор источников прерывания. Параметр принимает любую совокупность значений из <a href="#">Маски источников прерываний</a> .

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 681

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 657

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 <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Status	Статус прерывания. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_cap.c строка 616

Перекрестные ссылки IS\_CAP\_ALL\_PERIPH и IS\_CAP\_IT\_SOURCE.

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 <a href="#">Маски источников прерываний</a> .

## Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_cap.c строка 643

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 <a href="#">CAP_PWM_Init_TypeDef</a> , 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\_Init\_TypeDef::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 <a href="#">CAP_PWM_Init_TypeDef</a> , 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_Init_↔ Struct	Указатель на структуру типа <a href="#">CAP_Init_TypeDef</a> , 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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .

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](mailto: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 функций 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 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.

### 8.9.1 Подробное описание

Файл содержит реализацию всех функции для работы с DMA.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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 <a href="#">DMA_Channel_TypeDef</a> , 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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_ChannelInitStruct	Указатель на структуру типа <a href="#">DMA_ChannelInit_TypeDef</a> , 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 <a href="#">DMA_ChannelInit_TypeDef</a> , 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"> <li>• ОК - ошибок не было;</li> <li>• ERROR - произошла ошибка.</li> </ul>
--------	---

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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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_InitStruct	Указатель на структуру типа <a href="#">DMA_Init_TypeDef</a> , 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 <a href="#">FunctionalState</a> .
-------	---

## 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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

DMA_↔Protection	Структура, содержащая конфигурацию защиты. Параметр принимает структуру типа <a href="#">DMA_Protect_TypeDef</a> .
-----------------	--

## 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

DMA_Channel	Выбор канала. Параметр принимает любую совокупность масок из <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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

State	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_InitStruct	Указатель на структуру типа <a href="#">DMA_Init_TypeDef</a> , 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.

8.9.2.18 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 <a href="#">Маски каналов по номеру</a> .
-------------	--

## 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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"> <li>• ENABLE - поддерживаются SREQ (как и блочные BREQ);</li> <li>• DISABLE - поддерживаются только блочные запросы BREQ.</li> </ul>
--------	---

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\\_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 DMA_Channel_UART0_TX DMA_Channel_0`
- `#define DMA_Channel_UART1_TX DMA_Channel_1`
- `#define DMA_Channel_UART2_TX DMA_Channel_2`
- `#define DMA_Channel_UART3_TX DMA_Channel_3`
- `#define DMA_Channel_UART0_RX DMA_Channel_4`
- `#define DMA_Channel_UART1_RX DMA_Channel_5`
- `#define DMA_Channel_UART2_RX DMA_Channel_6`
- `#define DMA_Channel_UART3_RX DMA_Channel_7`
- `#define DMA_Channel_ADCSEQ0 DMA_Channel_8`
- `#define DMA_Channel_ADCSEQ1 DMA_Channel_9`
- `#define DMA_Channel_ADCSEQ2 DMA_Channel_10`
- `#define DMA_Channel_ADCSEQ3 DMA_Channel_11`
- `#define DMA_Channel_ADCSEQ4 DMA_Channel_12`
- `#define DMA_Channel_ADCSEQ5 DMA_Channel_13`
- `#define DMA_Channel_ADCSEQ6 DMA_Channel_14`
- `#define DMA_Channel_ADCSEQ7 DMA_Channel_15`
- `#define DMA_Channel_SPI0_TX DMA_Channel_16`
- `#define DMA_Channel_SPI1_TX DMA_Channel_17`
- `#define DMA_Channel_SPI2_TX DMA_Channel_18`
- `#define DMA_Channel_SPI3_TX DMA_Channel_19`
- `#define DMA_Channel_SPI0_RX DMA_Channel_20`
- `#define DMA_Channel_SPI1_RX DMA_Channel_21`
- `#define DMA_Channel_SPI2_RX DMA_Channel_22`
- `#define DMA_Channel_SPI3_RX DMA_Channel_23`
- `#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](mailto: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 АHB-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 АHB-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 АНВ-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 93

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

Канал DMA 1

См. определение в файле niietcm4\_dma.h строка 94

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

Канал DMA 10

См. определение в файле niietcm4\_dma.h строка 103

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

Канал DMA 11

См. определение в файле niietcm4\_dma.h строка 104

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

Канал DMA 12

См. определение в файле niietcm4\_dma.h строка 105

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

Канал DMA 13

См. определение в файле niietcm4\_dma.h строка 106

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

Канал DMA 14

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

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

Канал DMA 15

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

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

Канал DMA 16

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

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

Канал DMA 17

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

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

Канал DMA 18

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

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

Канал DMA 19

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

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

Канал DMA 2

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

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

Канал DMA 20

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

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

Канал DMA 21

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

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

Канал DMA 22

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

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

Канал DMA 23

См. определение в файле niietcm4\_dma.h строка 116

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

Канал DMA 3

См. определение в файле niietcm4\_dma.h строка 96

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

Канал DMA 4

См. определение в файле niietcm4\_dma.h строка 97

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

Канал DMA 5

См. определение в файле niietcm4\_dma.h строка 98

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

Канал DMA 6

См. определение в файле niietcm4\_dma.h строка 99

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

Канал DMA 7

См. определение в файле niietcm4\_dma.h строка 100

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

Канал DMA 8

См. определение в файле niietcm4\_dma.h строка 101

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

Канал DMA 9

См. определение в файле niietcm4\_dma.h строка 102

8.10.2.45 `#define DMA_Channel_ADCSEQ0 DMA_Channel_8`

Канал DMA секвенсора 0 АЦП

См. определение в файле niietcm4\_dma.h строка 134

8.10.2.46 `#define DMA_Channel_ADCSEQ1 DMA_Channel_9`

Канал DMA секвенсора 1 АЦП

См. определение в файле niietcm4\_dma.h строка 135

8.10.2.47 `#define DMA_Channel_ADCSEQ2 DMA_Channel_10`

Канал DMA секвенсора 2 АЦП

См. определение в файле niietcm4\_dma.h строка 136

8.10.2.48 `#define DMA_Channel_ADCSEQ3 DMA_Channel_11`

Канал DMA секвенсора 3 АЦП

См. определение в файле niietcm4\_dma.h строка 137

8.10.2.49 `#define DMA_Channel_ADCSEQ4 DMA_Channel_12`

Канал DMA секвенсора 4 АЦП

См. определение в файле niietcm4\_dma.h строка 138

8.10.2.50 `#define DMA_Channel_ADCSEQ5 DMA_Channel_13`

Канал DMA секвенсора 5 АЦП

См. определение в файле niietcm4\_dma.h строка 139

8.10.2.51 `#define DMA_Channel_ADCSEQ6 DMA_Channel_14`

Канал DMA секвенсора 6 АЦП

См. определение в файле niietcm4\_dma.h строка 140

8.10.2.52 `#define DMA_Channel_ADCSEQ7 DMA_Channel_15`

Канал DMA секвенсора 7 АЦП

См. определение в файле niietcm4\_dma.h строка 141

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 DMA_Channel_20`

Канал DMA по приему от SPI0

См. определение в файле niietcm4\_dma.h строка 146

8.10.2.55 `#define DMA_Channel_SPI0_TX DMA_Channel_16`

Канал DMA по передаче от SPI0

См. определение в файле niietcm4\_dma.h строка 142



8.10.2.56 `#define DMA_Channel_SPI1_RX DMA_Channel_21`

Канал DMA по приему от SPI1

См. определение в файле niietcm4\_dma.h строка 147

8.10.2.57 `#define DMA_Channel_SPI1_TX DMA_Channel_17`

Канал DMA по передаче от SPI1

См. определение в файле niietcm4\_dma.h строка 143

8.10.2.58 `#define DMA_Channel_SPI2_RX DMA_Channel_22`

Канал DMA по приему от SPI2

См. определение в файле niietcm4\_dma.h строка 148

8.10.2.59 `#define DMA_Channel_SPI2_TX DMA_Channel_18`

Канал DMA по передаче от SPI2

См. определение в файле niietcm4\_dma.h строка 144

8.10.2.60 `#define DMA_Channel_SPI3_RX DMA_Channel_23`

Канал DMA по приему от SPI3

См. определение в файле niietcm4\_dma.h строка 149

8.10.2.61 `#define DMA_Channel_SPI3_TX DMA_Channel_19`

Канал DMA по передаче от SPI3

См. определение в файле niietcm4\_dma.h строка 145

8.10.2.62 `#define DMA_Channel_UART0_RX DMA_Channel_4`

Канал DMA по приему от UART0

См. определение в файле niietcm4\_dma.h строка 130

8.10.2.63 `#define DMA_Channel_UART0_TX DMA_Channel_0`

Канал DMA по передаче от UART0

См. определение в файле niietcm4\_dma.h строка 126

8.10.2.64 `#define DMA_Channel_UART1_RX DMA_Channel_5`

Канал DMA по приему от UART1

См. определение в файле niietcm4\_dma.h строка 131

8.10.2.65 `#define DMA_Channel_UART1_TX DMA_Channel_1`

Канал DMA по передаче от UART1

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

8.10.2.66 `#define DMA_Channel_UART2_RX DMA_Channel_6`

Канал DMA по приему от UART2

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

8.10.2.67 `#define DMA_Channel_UART2_TX DMA_Channel_2`

Канал DMA по передаче от UART2

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

8.10.2.68 `#define DMA_Channel_UART3_RX DMA_Channel_7`

Канал DMA по приему от UART3

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

8.10.2.69 `#define DMA_Channel_UART3_TX DMA_Channel_3`

Канал DMA по передаче от UART3

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

8.10.2.70 `#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()`.

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)) || \
  ((CHANNEL) == (DMA_Channel_1)) || \
  ((CHANNEL) == (DMA_Channel_2)) || \
  ((CHANNEL) == (DMA_Channel_3)) || \
```

```

((CHANNEL) == (DMA_Channel_4)) || \
((CHANNEL) == (DMA_Channel_5)) || \
((CHANNEL) == (DMA_Channel_6)) || \
((CHANNEL) == (DMA_Channel_7)) || \
((CHANNEL) == (DMA_Channel_8)) || \
((CHANNEL) == (DMA_Channel_9)) || \
((CHANNEL) == (DMA_Channel_10)) || \
((CHANNEL) == (DMA_Channel_11)) || \
((CHANNEL) == (DMA_Channel_12)) || \
((CHANNEL) == (DMA_Channel_13)) || \
((CHANNEL) == (DMA_Channel_14)) || \
((CHANNEL) == (DMA_Channel_15)) || \
((CHANNEL) == (DMA_Channel_16)) || \
((CHANNEL) == (DMA_Channel_17)) || \
((CHANNEL) == (DMA_Channel_18)) || \
((CHANNEL) == (DMA_Channel_10)) || \
((CHANNEL) == (DMA_Channel_20)) || \
((CHANNEL) == (DMA_Channel_21)) || \
((CHANNEL) == (DMA_Channel_22)) || \
((CHANNEL) == (DMA_Channel_23))

```

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 <a href="#">DMA_Channel_TypeDef</a> , 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">DMA_ChannelInit_TypeDef</a> , 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.

8.10.4.5 void DMA\_ChannelStructInit ( DMA\_ChannelInit\_TypeDef \* DMA\_ChannelInitStruct )

Заполнение каждого члена структуры DMA\_ChannelInitStruct значениями по умолчанию.

Аргументы

DMA_ChannelInitStruct	Указатель на структуру типа <a href="#">DMA_ChannelInit_TypeDef</a> , 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.

8.10.4.6 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"> <li>• OK - ошибок не было;</li> <li>• ERROR - произошла ошибка.</li> </ul>
--------	---

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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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_Init↔ Struct	Указатель на структуру типа <a href="#">DMA_Init_TypeDef</a> , 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 <a href="#">FunctionalState</a> .
-------	---

## 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">DMA_Protect_TypeDef</a> .
---------------------	--

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().

8.10.4.15 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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

DMA_Init↵ Struct	Указатель на структуру типа <a href="#">DMA_Init_TypeDef</a> , 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.

8.10.4.18 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 <a href="#">Маски каналов по номеру</a> .
-------------	--

## 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 <a href="#">Маски каналов по номеру</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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

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](mailto: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 <a href="#">EXTMEM_Init_TypeDef</a> , 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 <a href="#">EXTMEM_Init_TypeDef</a> , 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@niiet.ru](mailto: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"

## 8.12.2 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) || \
    ((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()`.

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 <a href="#">EXTMEM_Init_TypeDef</a> , 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 <a href="#">EXTMEM_Init_TypeDef</a> , 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 функций 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)`  
Осуществляет выбор альтернативной функции вывода GPIOx.
- `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 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.

### 8.13.1 Подробное описание

Файл содержит реализацию всех функции для работы с модулями GPIO.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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_AltFuncConfig ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc )`

Осуществляет выбор альтернативной функции вывода GPIOx.

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.
GPIO_AltFunc	Выбор номера альтернативной функции пина. Параметр принимает любое значение из <a href="#">GPIO_AltFunc_TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 278

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH, IS\_GPIO\_ALT\_FUNC и IS\_GPIO\_PIN.

Используется в GPIO\_Init().

8.13.3.2 `void GPIO_ClearBits ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin )`

Сброс 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 635

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.3 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.

8.13.3.4 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 <a href="#">GPIO_Init_TypeDef</a> , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 340

Перекрестные ссылки GPIO\_Init\_TypeDef::GPIO\_AltFunc, GPIO\_AltFuncConfig(), 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().

8.13.3.5 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 922

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.6 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 <a href="#">GPIO_IntType_TypeDef</a> .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из <a href="#">GPIO_IntPol_TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 885

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.

8.13.3.7 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 947

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.8 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 762

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.9 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 <a href="#">GPIO_QualMode_TypeDef</a> .
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 673

Перекрестные ссылки GPIO\_QualMode\_3sample, GPIO\_QualMode\_6sample, IS\_GPIO\_ALL\_PERIPH, IS\_GPIO\_PIN, IS\_GPIO\_QUAL\_MODE и IS\_GPIO\_QUAL\_PERIOD.

8.13.3.10 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 512

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

8.13.3.11 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 486

Перекрестные ссылки Bit\_CLEAR, Bit\_SET, IS\_GET\_GPIO\_PIN и IS\_GPIO\_ALL\_PERIPH.

8.13.3.12 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 527

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

8.13.3.13 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 618

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.14 void GPIO\_StructInit ( GPIO\_Init\_TypeDef \* GPIO\_InitStruct )

Заполнение каждого члена структуры GPIO\_InitStruct значениями по умолчанию.

Аргументы

GPIO_Init↔ Struct	Указатель на структуру типа <a href="#">GPIO_Init_TypeDef</a> , которую необходимо проинициализировать.
----------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 465

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

IO\_Pin\_All, GPIO\_Init\_TypeDef::GPIO\_PullUp и GPIO\_PullUp\_Dis.

Используется в RCC\_SysClkDiv2Out().

8.13.3.15 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 823

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.16 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 652

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.13.3.17 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 579

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

8.13.3.18 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 <a href="#">BitAction</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 555

Перекрестные ссылки [Bit\\_CLEAR](#), [Bit\\_SET](#), [IS\\_GET\\_GPIO\\_PIN](#), [IS\\_GPIO\\_ALL\\_PERIPH](#) и [IS\\_GPIO\\_BIT\\_ACTION](#).

8.13.3.19 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 595

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)  
Осуществляет выбор альтернативной функции вывода GPIOx.
- 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](mailto: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_AltFuncConfig()` и `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 niietcm4\_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 niietcm4\_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 niietcm4\_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 niietcm4\_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 niietcm4\_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_AltFuncConfig ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,  
GPIO_AltFunc_TypeDef GPIO_AltFunc )`

Осуществляет выбор альтернативной функции вывода `GPIOx`.

## 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.
GPIO_AltFunc	Выбор номера альтернативной функции пина. Параметр принимает любое значение из <a href="#">GPIO_AltFunc_TypeDef</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле `niietcm4_gpio.c` строка 278

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`, `IS_GPIO_ALT_FUNC` и `IS_GPIO_PIN`.

Используется в `GPIO_Init()`.

8.14.4.2 `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 635

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.14.4.3 `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_GPIOSEEx_Reset_Value`, `GPIO_Regs_A_C_E_G_Mask`, `GPIO_Regs_B_D_F_H_Mask` и `IS_GPIO_ALL_PERIPH`.

8.14.4.4 `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_Init↵ Struct	Указатель на структуру типа <a href="#">GPIO_Init_TypeDef</a> , которая содержит конфигурационную информацию.

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 340

Перекрестные ссылки [GPIO\\_Init\\_TypeDef::GPIO\\_AltFunc](#), [GPIO\\_AltFuncConfig\(\)](#), [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](#), [GP↵IO\\_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::GP↵IO\\_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\(\)](#).

```
8.14.4.5 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 <a href="#">GPIO_Pin_x</a> , где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 922

Перекрестные ссылки [IS\\_FUNCTIONAL\\_STATE](#), [IS\\_GPIO\\_ALL\\_PERIPH](#) и [IS\\_GPIO\\_PIN](#).

```
8.14.4.6 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 <a href="#">GPIO_Pin_x</a> , где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из <a href="#">GPIO_IntType_TypeDef</a> .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из <a href="#">GPIO_IntPol_TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 885

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.

8.14.4.7 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 947

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.14.4.8 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 762

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.14.4.9 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 <a href="#">GPIO_↵_QualMode_TypeDef</a> .
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 673

Перекрестные ссылки GPIO\_QualMode\_3sample, GPIO\_QualMode\_6sample, IS\_GPIO\_ALL\_PERIPH, IS\_GPIO\_PIN, IS\_GPIO\_QUAL\_MODE и IS\_GPIO\_QUAL\_PERIOD.

8.14.4.10 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 512

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

8.14.4.11 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 486

Перекрестные ссылки Bit\_CLEAR, Bit\_SET, IS\_GET\_GPIO\_PIN и IS\_GPIO\_ALL\_PERIPH.

8.14.4.12 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 527

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

8.14.4.13 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 618

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.14.4.14 void GPIO\_StructInit ( GPIO\_Init\_TypeDef \* GPIO\_InitStruct )

Заполнение каждого члена структуры GPIO\_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа <a href="#">GPIO_Init_TypeDef</a> , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 465

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().

8.14.4.15 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 <a href="#">FunctionalState</a> .
-------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_gpio.c строка 823

Перекрестные ссылки IS\_FUNCTIONAL\_STATE, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.14.4.16 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 652

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_PIN.

8.14.4.17 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 579

Перекрестные ссылки IS\_GPIO\_ALL\_PERIPH.

8.14.4.18 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 <a href="#">BitAction</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_gpio.c строка 555

Перекрестные ссылки Bit\_CLEAR, Bit\_SET, IS\_GET\_GPIO\_PIN, IS\_GPIO\_ALL\_PERIPH и IS\_GPIO\_BIT\_ACTION.

8.14.4.19 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 595

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 частой 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](mailto: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 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"

### 8.15.2 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 <a href="#">RCC_ADCClk_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">RCC_ADCClk_TypeDef</a> .
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 <a href="#">FunctionalState</a> .

## 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 <a href="#">RCC_PeriphClk_↔TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">RCC_↔PeriphRst_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## 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 <a href="#">RCC_PLLRef_TypeDef</a> .
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_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, 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\_PLLNR\_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:

$$F_{REF} = F_{IN} / (2 \times NR),$$

а частота  $F_{VCO}$  вычисляется по формуле:

$$F_{VCO} = F_{IN} \times (NF / NR)$$

## Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа <a href="#">RCC_PLLInit_TypeDef</a> , 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 <a href="#">FunctionalState</a> .
-------	---

## 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 <a href="#">RCC_PLLInit_TypeDef</a> , 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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">FunctionalState</a> .

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 <a href="#">RCC_SPIClk_TypeDef</a> .

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 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"> <li>• ENABLE - переводит H[0] в выход включенной альтернативной функцией 2.</li> <li>• DISABLE - переводит H[0] в состояние по умолчанию.</li> </ul>
-------	--

## 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 <a href="#">RCC_SysClk_TypeDef</a> .
------------	--

## 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 <a href="#">RCC_SysClk_TypeDef</a>
----------	---------------------------------------

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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .

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 <a href="#">RCC_UARTClk_TypeDef</a> .

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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">RCC_USBClk_TypeDef</a> .
RCC_USB<math>\leftrightarrow</math>Freq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из <a href="#">RCC_USBFreq_TypeDef</a> .

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)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)↵  
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](mailto: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 <a href="#">RCC_ADCClk_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">RCC_ADCClk_TypeDef</a> .
------------	---



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 <a href="#">FunctionalState</a> .

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 <a href="#">RCC_PeriphClk_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">RCC_PeriphRst_TypeDef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

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 <a href="#">RCC_PLLRef_TypeDef</a> .
<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 <a href="#">RCC_PLLInit_TypeDef</a> , 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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">RCC_PLLInit_TypeDef</a> , 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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .
----------	--

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 <a href="#">RCC_SPIClk_TypeDef</a> .

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"> <li>• ENABLE - переводит H[0] в выход включенной альтернативной функцией 2.</li> <li>• DISABLE - переводит H[0] в состояние по умолчанию.</li> </ul>
-------	--

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 <a href="#">RCC_SysClk_Type</a> ↵ <a href="#">Def</a> .
------------	--

## 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 <a href="#">RCC_SysClk_TypeDef</a>
----------	---------------------------------------

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 <a href="#">FunctionalState</a> .

## 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 <a href="#">FunctionalState</a> ↵.

## 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 <a href="#">RCC_UARTClk_TypeDef</a> .

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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">RCC_USBClk_TypeDef</a> .
RCC_USB↔Freq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из <a href="#">RCC_USBFreq_TypeDef</a> .

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](mailto:dykhno@niiet.ru)
- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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)0x10)`, `RTC_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.

## 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](mailto:dykhno@niiet.ru)
- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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 функций для работы с таймерами

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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 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"

### 8.19.2 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 <a href="#">FunctionalState</a> .

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 <a href="#">TIMER_Ext↔ Input_TypeDef</a> .

## 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 <a href="#">FunctionalState</a> .

## 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](mailto: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 <a href="#">FunctionalState</a> .

## 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 <a href="#">TIMER_ExtInput_TypeDef</a> .

## 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 <a href="#">FunctionalState</a> .

## 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 ←  
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, uint32\_t UART\_Flag)  
Запрос состояния выбранного флага.
- [FlagStatus](#) [UART\\_ErrorStatus](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_Error)  
Запрос состояния выбранного флага ошибки.
- void [UART\\_ErrorStatusClear](#) (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_Error)  
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 генерироваться прерывание.

- void `UART_ITCmd` (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource, [FunctionalState](#) State)

Маскирование выбранных прерываний.

- [FlagStatus](#) `UART_ITRawStatus` (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)

Запрос немаскированного состояния прерывания.

- [FlagStatus](#) `UART_ITMaskedStatus` (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_ITSource)

Запрос маскированного состояния прерывания.

- void `UART_ITStatusClear` (NT\_UART\_TypeDef \*UARTx, uint32\_t 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@niet.ru](mailto: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.



## 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.

Используется в UART\_Init().

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 <a href="#">FunctionalState</a> .

## 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 <a href="#">FunctionalState</a> .

## 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 <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 515

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 <a href="#">UART_Dir_Typedef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 533

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, uint32\_t UART\_Error )

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 313

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

8.21.2.8 void UART\_ErrorStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Error )

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 339

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

8.21.2.9 FlagStatus UART\_FlagStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Flag )

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флагов. Параметр принимает любую совокупность значений из <a href="#">Флаги работы UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 286

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_InitStruct	Указатель на структуру типа <a href="#">UART_Init_TypeDef</a> , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из <a href="#">OperationStatus</a> .
--------	---

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\_BaudRateDivConfig(), UART\_Init\_TypeDef::UART←\_ClkFreq, UART\_Init\_TypeDef::UART\_DataWidth, UART\_Init\_TypeDef::UART\_FIFOEn, UAR←\_T\_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, uint32\_t 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 <a href="#">Ис-точники прерываний UART</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 421

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 <a href="#">UART_Dir_Typedef</a> .
UART_FIF←_OLevel	Выбор порога. Параметр принимает любое значение из <a href="#">UART_FIFOLevel_←_TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 395

Перекрестные ссылки IS\_UART\_ALL\_PERIPH, IS\_UART\_DIR, IS\_UART\_FIFO\_LEVEL и U←\_ART\_Dir\_Rx.

8.21.2.13 FlagStatus UART\_ITMaskedStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_ITSource )

Запрос маскированного состояния прерывания.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 472

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

8.21.2.14 FlagStatus UART\_ITRawStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_ITSource )

Запрос немаскированного состояния прерывания.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любую совокупность значений из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 445

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

8.21.2.15 void UART\_ITStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 любую совокупность значений из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 498

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 <a href="#">UART_ModemInit_TypeDef</a> , которая содержит конфигурационную информацию.
---------------------------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 355

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.

```
8.21.2.17 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 <a href="#">UART_ModemInit_TypeDef</a> , которую необходимо проинициализировать.
---------------------------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 376

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 270

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 249

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_InitStruct	Указатель на структуру типа <a href="#">UART_Init_TypeDef</a> , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 228

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.

## 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\\_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`.
- `#define UART_ITSource_ChangeRI ((uint32_t)0x00000001)`
- `#define UART_ITSource_ChangeCTS ((uint32_t)0x00000002)`
- `#define UART_ITSource_ChangeDCD ((uint32_t)0x00000004)`
- `#define UART_ITSource_ChangeDSR ((uint32_t)0x00000008)`
- `#define UART_ITSource_RxFIFOLevel ((uint32_t)0x00000010)`
- `#define UART_ITSource_TxFIFOLevel ((uint32_t)0x00000020)`
- `#define UART_ITSource_RecieveTimeout ((uint32_t)0x00000040)`
- `#define UART_ITSource_ErrorFrame ((uint32_t)0x00000080)`
- `#define UART_ITSource_ErrorParity ((uint32_t)0x00000100)`
- `#define UART_ITSource_ErrorBreak ((uint32_t)0x00000200)`
- `#define UART_ITSource_ErrorOverflow ((uint32_t)0x00000400)`
- `#define UART_ITSource_All ((uint32_t)0x000007FF)`
- `#define IS_UART_IT_SOURCE(IT_SOURCE) (((IT_SOURCE) & ~UART_ITSource_All) == 0)`  
Макрос проверки номеров источников прерываний на попадание в допустимый диапазон.
- `#define UART_Flag_InvCTS ((uint32_t)0x00000001)`
- `#define UART_Flag_InvDSR ((uint32_t)0x00000002)`
- `#define UART_Flag_InvDCD ((uint32_t)0x00000004)`
- `#define UART_Flag_Busy ((uint32_t)0x00000008)`
- `#define UART_Flag_RxFIFOEmpty ((uint32_t)0x00000010)`
- `#define UART_Flag_TxFIFOFull ((uint32_t)0x00000020)`
- `#define UART_Flag_RxFIFOFull ((uint32_t)0x00000040)`
- `#define UART_Flag_TxFIFOEmpty ((uint32_t)0x00000080)`
- `#define UART_Flag_InvRI ((uint32_t)0x00000100)`
- `#define UART_Flag_All ((uint32_t)0x000001FF)`
- `#define IS_UART_FLAG(FLAG) (((FLAG) & ~UART_Flag_All) == 0)`  
Макрос проверки номеров флагов на попадание в допустимый диапазон.
- `#define UART_Error_Frame ((uint32_t)0x00000001)`
- `#define UART_Error_Parity ((uint32_t)0x00000002)`
- `#define UART_Error_Break ((uint32_t)0x00000004)`
- `#define UART_Error_Overflow ((uint32_t)0x00000008)`
- `#define UART_Error_All ((uint32_t)0x0000000F)`
- `#define IS_UART_ERROR(ERROR) (((ERROR) & ~UART_Error_All) == 0)`  
Макрос проверки номеров флагов ошибок на попадание в допустимый диапазон.

## Перечисления

- `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_ParityBit_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_FIFOLevel_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 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`, `uint32_t UART_Flag`)  
Запрос состояния выбранного флага.
- `FlagStatus` `UART_ErrorStatus` (`NT_UART_TypeDef *UARTx`, `uint32_t UART_Error`)  
Запрос состояния выбранного флага ошибки.
- void `UART_ErrorStatusClear` (`NT_UART_TypeDef *UARTx`, `uint32_t UART_Error`)  
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`, `uint32_t UART_ITSource`, `FunctionalState State`)  
Маскирование выбранных прерываний.
- `FlagStatus` `UART_ITRawStatus` (`NT_UART_TypeDef *UARTx`, `uint32_t UART_ITSource`)  
Запрос немаскированного состояния прерывания.

- `FlagStatus UART_ITMaskedStatus` (NT\_UART\_TypeDef \*UARTx, uint32\_t UART\_IT ← Source)  
Запрос маскированного состояния прерывания.
- `void UART_ITStatusClear` (NT\_UART\_TypeDef \*UARTx, uint32\_t 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](mailto: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 143

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 84

Используется в UART\_DMAMCmd() и UART\_ITFIFOLevelConfig().

8.22.2.3 #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 166

Используется в UART\_Init() и UART\_ITFIFOLevelConfig().

8.22.2.4 #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 121

Используется в UART\_Init().

8.22.2.5 #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 101

Используется в UART\_Init().

8.22.2.6 `#define UART_Error_All ((uint32_t)0x0000000F)`

Все флаги ошибок выбраны.

См. определение в файле niietcm4\_uart.h строка 283

8.22.2.7 `#define UART_Error_Break ((uint32_t)0x00000004)`

Флаг разрыва линии.

См. определение в файле niietcm4\_uart.h строка 281

8.22.2.8 `#define UART_Error_Frame ((uint32_t)0x00000001)`

Флаг ошибки в структуре кадра.

См. определение в файле niietcm4\_uart.h строка 279

8.22.2.9 `#define UART_Error_Overflow ((uint32_t)0x00000008)`

Флаг переполнения буфера приемника.

См. определение в файле niietcm4\_uart.h строка 282

8.22.2.10 `#define UART_Error_Parity ((uint32_t)0x00000002)`

Флаг ошибки контроля четности.

См. определение в файле niietcm4\_uart.h строка 280

8.22.2.11 `#define UART_Flag_All ((uint32_t)0x000001FF)`

Все флаги выбраны.

См. определение в файле niietcm4\_uart.h строка 263

8.22.2.12 `#define UART_Flag_Busy ((uint32_t)0x00000008)`

Флаг занятости блока UART.

См. определение в файле niietcm4\_uart.h строка 257

8.22.2.13 `#define UART_Flag_InvCTS ((uint32_t)0x00000001)`

Флаг инверсии сигнала на линии UART\_CTS.

См. определение в файле niietcm4\_uart.h строка 254

8.22.2.14 `#define UART_Flag_InvDCD ((uint32_t)0x00000004)`

Флаг инверсии сигнала на линии UART\_DSR.

См. определение в файле niietcm4\_uart.h строка 256

8.22.2.15 `#define UART_Flag_InvDSR ((uint32_t)0x00000002)`

Флаг инверсии сигнала на линии UART\_DSR.

См. определение в файле niietcm4\_uart.h строка 255

8.22.2.16 #define UART\_Flag\_InvRI ((uint32\_t)0x00000100)

Флаг инверсии сигнала на линии UART\_RI.

См. определение в файле niietcm4\_uart.h строка 262

8.22.2.17 #define UART\_Flag\_RxFIFOEmpty ((uint32\_t)0x00000010)

Флаг пустоты буфера приемника.

См. определение в файле niietcm4\_uart.h строка 258

8.22.2.18 #define UART\_Flag\_RxFIFOFull ((uint32\_t)0x00000040)

Флаг заполнения буфера приемника.

См. определение в файле niietcm4\_uart.h строка 260

8.22.2.19 #define UART\_Flag\_TxFIFOEmpty ((uint32\_t)0x00000080)

Флаг пустоты буфера передатчика.

См. определение в файле niietcm4\_uart.h строка 261

8.22.2.20 #define UART\_Flag\_TxFIFOFull ((uint32\_t)0x00000020)

Флаг заполнения буфера передатчика.

См. определение в файле niietcm4\_uart.h строка 259

8.22.2.21 #define UART\_ITSource\_All ((uint32\_t)0x000007FF)

Все источники выбраны.

См. определение в файле niietcm4\_uart.h строка 238

8.22.2.22 #define UART\_ITSource\_ChangeCTS ((uint32\_t)0x00000002)

Изменение состояния линии UART\_CTS

См. определение в файле niietcm4\_uart.h строка 228

8.22.2.23 #define UART\_ITSource\_ChangeDCD ((uint32\_t)0x00000004)

Изменение состояния линии UART\_DCD

См. определение в файле niietcm4\_uart.h строка 229

8.22.2.24 #define UART\_ITSource\_ChangeDSR ((uint32\_t)0x00000008)

Изменение состояния линии UART\_DSR

См. определение в файле niietcm4\_uart.h строка 230

8.22.2.25 `#define UART_ITSource_ChangeRI ((uint32_t)0x00000001)`

Изменение состояния линии UART\_RI

См. определение в файле niietcm4\_uart.h строка 227

8.22.2.26 `#define UART_ITSource_ErrorBreak ((uint32_t)0x00000200)`

Разрыв линии

См. определение в файле niietcm4\_uart.h строка 236

8.22.2.27 `#define UART_ITSource_ErrorFrame ((uint32_t)0x00000080)`

Ошибка в структуре кадра

См. определение в файле niietcm4\_uart.h строка 234

8.22.2.28 `#define UART_ITSource_ErrorOverflow ((uint32_t)0x00000400)`

Переполнение буфера приемника

См. определение в файле niietcm4\_uart.h строка 237

8.22.2.29 `#define UART_ITSource_ErrorParity ((uint32_t)0x00000100)`

Ошибка контроля четности

См. определение в файле niietcm4\_uart.h строка 235

8.22.2.30 `#define UART_ITSource_RecieveTimeout ((uint32_t)0x00000040)`

Таймаут приема данных

См. определение в файле niietcm4\_uart.h строка 233

8.22.2.31 `#define UART_ITSource_RxFIFOLevel ((uint32_t)0x00000010)`

Порог переполнения буфера приемника

См. определение в файле niietcm4\_uart.h строка 231

8.22.2.32 `#define UART_ITSource_TxFIFOLevel ((uint32_t)0x00000020)`

Порог опустошения буфера передатчика

См. определение в файле niietcm4\_uart.h строка 232

## 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 131

#### 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 74

#### 8.22.3.3 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 153

#### 8.22.3.4 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 108

#### 8.22.3.5 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 91



## 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.

Используется в UART\_Init().

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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .

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 <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 515

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 <a href="#">UART_Dir_Typedef</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 533

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, uint32_t UART_Error )
```

Запрос состояния выбранного флага ошибки.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 313

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

8.22.4.8 void UART\_ErrorStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Error )

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флагов ошибки. Параметр принимает любую совокупность значений из <a href="#">Ошибки приемника UART</a> .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 339

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_ERROR.

8.22.4.9 FlagStatus UART\_FlagStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_Flag )

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флагов. Параметр принимает любую совокупность значений из <a href="#">Флаги работы UART</a> .

Возвращаемые значения

Status	Состояние флага. Если выбрано несколько флагов, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 286

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 <a href="#">UART_Init_TypeDef</a> , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из <a href="#">OperationStatus</a> .
--------	---

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\_BaudRateDivConfig(), UART\_Init\_TypeDef::UART\_←  
ClkFreq, UART\_Init\_TypeDef::UART\_DataWidth, UART\_Init\_TypeDef::UART\_FIFOEn, UAR\_←  
T\_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.

8.22.4.11 void UART\_ITCmd ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 <a href="#">Ис-точники прерываний UART</a> .
State	Выбор состояния. Параметр принимает любое значение из <a href="#">FunctionalState</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 421

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 генери-  
роваться 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 <a href="#">UART_Dir_Typedef</a> .
UART_FIF_← OLevel	Выбор порога. Параметр принимает любое значение из <a href="#">UART_FIFOLevel_←TypeDef</a> .

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 395

Перекрестные ссылки IS\_UART\_ALL\_PERIPH, IS\_UART\_DIR, IS\_UART\_FIFO\_LEVEL и U\_←  
ART\_Dir\_Rx.

8.22.4.13 FlagStatus UART\_ITMaskedStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t  
UART\_ITSource )

Запрос маскированного состояния прерывания.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 472

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

8.22.4.14 FlagStatus UART\_ITRawStatus ( NT\_UART\_TypeDef \* UARTx, uint32\_t UART\_ITSource )

Запрос немаскированного состояния прерывания.

## Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любую совокупность значений из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Status	Состояние флага. Если выбрано несколько прерываний, то результат соответствует логическому ИЛИ их состояний.
--------	--

См. определение в файле niietcm4\_uart.c строка 445

Перекрестные ссылки IS\_UART\_ALL\_PERIPH и IS\_UART\_IT\_SOURCE.

8.22.4.15 void UART\_ITStatusClear ( NT\_UART\_TypeDef \* UARTx, uint32\_t 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 любую совокупность значений из <a href="#">Источники прерываний UART</a> .

## Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4\_uart.c строка 498

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 <a href="#">UART_ModemInit_TypeDef</a> , которая содержит конфигурационную информацию.
---------------------------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 355

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.

```
8.22.4.17 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 <a href="#">UART_ModemInit_TypeDef</a> , которую необходимо проинициализировать.
---------------------------------	--

## Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 376

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 270

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 249

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_InitStruct	Указатель на структуру типа <a href="#">UART_Init_TypeDef</a> , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет
-----

См. определение в файле niietcm4\_uart.c строка 228

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.

## 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 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.

### 8.23.1 Подробное описание

Файл содержит реализацию всех функции для работы с пользовательской флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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 <a href="#">FunctionalState</a> .
-------	---

## 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 флэш.

## Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из <a href="#">USERFLASH_Status_TypeDef</a> .
--------	---

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 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.

## 8.24 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\_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.

## 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.
- 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.
- 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.

### 8.24.1 Подробное описание

Файл содержит все прототипы функций для пользовательской флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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\_BYTE↵

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\_BYTE↵  
S\*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 <a href="#">FunctionalState</a> .
-------	---

## 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 <a href="#">USERFLASH_Status_TypeDef</a> .
--------	---

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 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.

## 8.24.4.10 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.

8.24.4.11 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 функции для работы со сторожевым таймером.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), [kolbov@niiet.ru](mailto: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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">FunctionalState</a> .
-------	---

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@niiet.ru](mailto: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 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.

© 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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">FunctionalState</a> .
-------	---

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 <a href="#">FunctionalState</a> .
-------	---

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`, 371
  - `CYCLE_CTRL`, 371
  - `DST_INC`, 371
  - `DST_PROT_BUFFERABLE`, 372
  - `DST_PROT_CACHEABLE`, 372
  - `DST_PROT_PRIVILEGED`, 372
  - `DST_SIZE`, 372
  - `N_MINUS_1`, 372
  - `NEXT_USEBURST`, 372
  - `R_POWER`, 372
  - `SRC_INC`, 373
  - `SRC_PROT_BUFFERABLE`, 373
  - `SRC_PROT_CACHEABLE`, 373
  - `SRC_PROT_PRIVILEGED`, 373
  - `SRC_SIZE`, 373
- Битовые операции, 155
  - `GPIO_ClearBits`, 155
  - `GPIO_SetBits`, 155
  - `GPIO_ToggleBits`, 155
- 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, 151
  - `GPIO_Read`, 151
  - `GPIO_ReadBit`, 151
  - `GPIO_ReadMask`, 151
- Чтение и запись, 150
- Драйвер периферии, 363
- Фильтрация, 157
  - `GPIO_QualCmd`, 157
  - `GPIO_QualConfig`, 157
  - `GPIO_SyncCmd`, 158
- Флаги работы UART, 204
  - `UART_Flag_All`, 204
  - `UART_Flag_Busy`, 204
  - `UART_Flag_InvCTS`, 204
  - `UART_Flag_InvDCD`, 204
  - `UART_Flag_InvDSR`, 204
  - `UART_Flag_InvRI`, 204
  - `UART_Flag_RxFIFOEmpty`, 204
  - `UART_Flag_RxFIFOFull`, 205
  - `UART_Flag_TxFIFOEmpty`, 205
  - `UART_Flag_TxFIFOFull`, 205
- Функции, 37, 66, 78, 115, 133, 147, 170, 187, 190, 207, 224, 232
  - `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`, 133
  - `EXTMEM_Init`, 133
  - `EXTMEM_StructInit`, 133
  - `RCC_SysClkDiv2Out`, 170
  - `UART_BaudRateDivConfig`, 207
  - `UART_Break`, 207
  - `UART_Cmd`, 208
  - `USERFLASH_ITCmd`, 224
  - `USERFLASH_Init`, 224
  - `USERFLASH_OperationStatus`, 224
  - `USERFLASH_OperationStatusClear`, 225
- Инициализация, 43
- Информационная область флеш, 64, 70, 222, 228
  - `BOOTFLASH_INFO_PAGE_SIZE_BYT↔ES`, 64
  - `BOOTFLASH_INFO_PAGE_TOTAL`, 64
  - `BOOTFLASH_INFO_TOTAL_BYTES`, 64
  - `BOOTFLASH_Info_PageErase`, 70
  - `BOOTFLASH_Info_Write`, 70
  - `USERFLASH_INFO_PAGE_SIZE_BYT↔ES`, 222
  - `USERFLASH_INFO_PAGE_TOTAL`, 222
  - `USERFLASH_INFO_TOTAL_BYTES`, 222
  - `USERFLASH_Info_PageErase`, 228
  - `USERFLASH_Info_Read`, 228
  - `USERFLASH_Info_Write`, 228
- Инициализация и деинициализация, 148, 209

- GPIO\_AltFuncConfig, 148
- GPIO\_DeInit, 148
- GPIO\_Init, 148
- GPIO\_StructInit, 149
- UART\_DeInit, 209
- UART\_Init, 209
- UART\_StructInit, 209
- Инициализация каналов DMA, 116
  - DMA\_ChannelDeInit, 116
  - DMA\_ChannelInit, 116
  - DMA\_ChannelStructInit, 117
- Инициализация контроллера DMA, 118
  - DMA\_DeInit, 118
  - DMA\_Init, 118
  - DMA\_StructInit, 118
- Источники прерываний UART, 202
  - UART\_ITSource\_All, 202
  - UART\_ITSource\_ChangeCTS, 202
  - UART\_ITSource\_ChangeDCD, 202
  - UART\_ITSource\_ChangeDSR, 202
  - UART\_ITSource\_ChangeRI, 202
  - UART\_ITSource\_ErrorBreak, 202
  - UART\_ITSource\_ErrorFrame, 203
  - UART\_ITSource\_ErrorOverflow, 203
  - UART\_ITSource\_ErrorParity, 203
  - UART\_ITSource\_RecieveTimeout, 203
  - UART\_ITSource\_RxFIFOLevel, 203
  - UART\_ITSource\_TxFIFOLevel, 203
- Конфигурация, 79, 191, 233
  - 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, 191
  - TIMER\_ExtInputConfig, 191
  - TIMER\_FreqConfig, 192
  - TIMER\_GetCounter, 192
  - TIMER\_GetReload, 192
  - TIMER\_PeriodConfig, 193
  - TIMER\_SetCounter, 193
  - TIMER\_SetReload, 193
  - WATCHDOG\_Cmd, 233
  - WATCHDOG\_GetCounter, 233
  - WATCHDOG\_GetReload, 233
  - WATCHDOG\_LockCmd, 234
  - WATCHDOG\_RstCmd, 234
  - WATCHDOG\_SetReload, 234
- Конфигурация PLL, 171
  - RCC\_PLLAutoConfig, 171
  - RCC\_PLLDeInit, 171
  - RCC\_PLLInit, 172
  - RCC\_PLLPowerDownCmd, 172
  - RCC\_PLLStructInit, 173
- Конфигурация контроллера DMA, 120
  - DMA\_BasePtrConfig, 120
  - DMA\_ChannelEnableCmd, 120
  - DMA\_HighPriorityCmd, 121
  - DMA\_MasterEnableCmd, 121
  - DMA\_PrmAltCmd, 121
  - DMA\_ProtectionConfig, 122
  - DMA\_ReqMaskCmd, 122
  - DMA\_SWRequestCmd, 122
  - DMA\_UseBurstCmd, 123
- Конфигурация прерываний, 54
- Конфигурация секвенсоров для DMA, 51
  - ADC\_SEQ\_DMAMaskCmd, 51
  - ADC\_SEQ\_DMAConfig, 51
  - ADC\_SEQ\_DMAErrorStatus, 51
  - ADC\_SEQ\_DMAErrorStatusClear, 53
- Константы, 15, 62, 75, 97, 126, 142, 161, 189, 201, 220, 231
  - RCC\_CLK\_CHANGE\_TIMEOUT, 161
  - RCC\_CLK\_PLL\_STABLE\_TIMEOUT, 161
- Макросы, 365
- Маски адреса, 127
  - EXTMEM\_CEMask\_Addr\_11, 127
  - EXTMEM\_CEMask\_Addr\_11\_19, 127
  - EXTMEM\_CEMask\_Addr\_12, 127
  - EXTMEM\_CEMask\_Addr\_13, 127
  - EXTMEM\_CEMask\_Addr\_14, 127
  - EXTMEM\_CEMask\_Addr\_15, 127
  - EXTMEM\_CEMask\_Addr\_16, 127
  - EXTMEM\_CEMask\_Addr\_17, 128
  - EXTMEM\_CEMask\_Addr\_18, 128
  - EXTMEM\_CEMask\_Addr\_19, 128
- 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, 76
  - CAP\_ITSource\_TimerEqPeriod, 77
  - CAP\_ITSource\_TimerOvf, 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, 106
  - DMA\_Channel\_ADCSEQ0, 106
  - DMA\_Channel\_ADCSEQ1, 106
  - DMA\_Channel\_ADCSEQ2, 106
  - DMA\_Channel\_ADCSEQ3, 106
  - DMA\_Channel\_ADCSEQ4, 106
  - DMA\_Channel\_ADCSEQ5, 107
  - DMA\_Channel\_ADCSEQ6, 107
  - DMA\_Channel\_ADCSEQ7, 107
  - DMA\_Channel\_SPI0\_RX, 107
  - DMA\_Channel\_SPI0\_TX, 107
  - DMA\_Channel\_SPI1\_RX, 107
  - DMA\_Channel\_SPI1\_TX, 107
  - DMA\_Channel\_SPI2\_RX, 107
  - DMA\_Channel\_SPI2\_TX, 107
  - DMA\_Channel\_SPI3\_RX, 108
  - DMA\_Channel\_SPI3\_TX, 108
  - DMA\_Channel\_UART0\_RX, 108
  - DMA\_Channel\_UART0\_TX, 108
  - DMA\_Channel\_UART1\_RX, 108
  - DMA\_Channel\_UART1\_TX, 108
  - DMA\_Channel\_UART2\_RX, 108
  - DMA\_Channel\_UART2\_TX, 108
  - DMA\_Channel\_UART3\_RX, 108
  - DMA\_Channel\_UART3\_TX, 108
- Маски каналов по номеру, 102
  - DMA\_Channel\_0, 102
  - DMA\_Channel\_1, 102
  - DMA\_Channel\_10, 102
  - DMA\_Channel\_11, 102
  - DMA\_Channel\_12, 102
  - DMA\_Channel\_13, 103
  - DMA\_Channel\_14, 103
  - DMA\_Channel\_15, 103
  - DMA\_Channel\_16, 103
  - DMA\_Channel\_17, 103
  - DMA\_Channel\_18, 103
  - DMA\_Channel\_19, 103
  - DMA\_Channel\_2, 103
  - DMA\_Channel\_20, 103
  - DMA\_Channel\_21, 104
  - DMA\_Channel\_22, 104
  - DMA\_Channel\_23, 104
  - DMA\_Channel\_3, 104
  - DMA\_Channel\_4, 104
  - DMA\_Channel\_5, 104
  - DMA\_Channel\_6, 104
  - DMA\_Channel\_7, 104
  - DMA\_Channel\_8, 104
  - DMA\_Channel\_9, 104
- Маски пинов, 143
  - GPIO\_Pin\_0, 143
  - GPIO\_Pin\_0\_3, 143
  - GPIO\_Pin\_0\_7, 143
  - GPIO\_Pin\_1, 143
  - GPIO\_Pin\_10, 144
  - GPIO\_Pin\_11, 144
  - GPIO\_Pin\_12, 144
  - GPIO\_Pin\_12\_15, 144
  - GPIO\_Pin\_13, 144
  - GPIO\_Pin\_14, 144
  - GPIO\_Pin\_15, 144
  - GPIO\_Pin\_2, 144
  - GPIO\_Pin\_3, 144
  - GPIO\_Pin\_4, 145
  - GPIO\_Pin\_4\_7, 145
  - GPIO\_Pin\_5, 145
  - GPIO\_Pin\_6, 145
  - GPIO\_Pin\_7, 145
  - GPIO\_Pin\_8, 145
  - GPIO\_Pin\_8\_11, 145
  - GPIO\_Pin\_8\_15, 145

- GPIO\_Pin\_9, 145
- GPIO\_Pin\_All, 145
- IS\_GET\_GPIO\_PIN, 146
- 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, 239, 282, 294, 300, 317
  - 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, 317
  - RCC\_PLL\_NF\_Reset\_Value, 317
  - RCC\_PLL\_NR\_Reset\_Value, 317
  - RCC\_PLL\_OD\_Reset\_Value, 317
- Настройка DMA, 219
  - UART\_DMABlkOnErrCmd, 219
  - UART\_DMACmd, 219
- Настройка драйвера, 364
  - EXT\_OSC\_VALUE, 364
  - INT\_OSC\_VALUE, 364
- Основная область флеш, 63, 68, 221, 226
  - BOOTFLASH\_FullErase, 68
  - BOOTFLASH\_PAGE\_SIZE\_BYTES, 63
  - BOOTFLASH\_PAGE\_TOTAL, 63
  - BOOTFLASH\_PageErase, 68
  - BOOTFLASH\_TOTAL\_BYTES, 63
  - BOOTFLASH\_Write, 68
  - USERFLASH\_FullErase, 226
  - USERFLASH\_PAGE\_SIZE\_BYTES, 221
  - USERFLASH\_PAGE\_TOTAL, 221
  - USERFLASH\_PageErase, 226
  - USERFLASH\_Read, 226
  - USERFLASH\_TOTAL\_BYTES, 221
  - USERFLASH\_Write, 227
- Ошибки приемника UART, 206
  - UART\_Error\_All, 206
  - UART\_Error\_Break, 206
  - UART\_Error\_Frame, 206
  - UART\_Error\_Overflow, 206
  - UART\_Error\_Parity, 206
- Периферия, 369
- Прерывания, 94, 159, 195, 216, 235
  - CAP\_ITCmd, 94
  - CAP\_ITForceCmd, 94
  - CAP\_ITPendClear, 95
  - CAP\_ITPendStatus, 95
  - CAP\_ITStatus, 95
  - CAP\_ITStatusClear, 95
  - GPIO\_ITCmd, 159
  - GPIO\_ITConfig, 159
  - GPIO\_ITStatusClear, 159
  - TIMER\_ITCmd, 195
  - TIMER\_ITStatus, 195
  - TIMER\_ITStatusClear, 195
  - UART\_ITCmd, 216
  - UART\_ITFIFOLevelConfig, 216

- UART\_ITMaskedStatus, [217](#)
- UART\_ITRawStatus, [217](#)
- UART\_ITStatusClear, [217](#)
- WATCHDOG\_ITMaskedStatus, [235](#)
- WATCHDOG\_ITRawStatus, [235](#)
- WATCHDOG\_ITStatusClear, [235](#)
- Прием и передача, [212](#)
  - UART\_ErrorStatus, [212](#)
  - UART\_ErrorStatusClear, [212](#)
  - UART\_FlagStatus, [212](#)
  - UART\_RecieveData, [214](#)
  - UART\_SendData, [214](#)
- Приватные данные, [237](#), [257](#), [263](#), [280](#), [292](#), [298](#), [315](#), [328](#), [331](#), [338](#), [351](#), [358](#)
- Приватные функции, [240](#), [259](#), [265](#), [283](#), [295](#), [305](#), [318](#), [329](#), [333](#), [340](#), [353](#), [360](#)
  - ADC\_Cmd, [242](#)
  - ADC\_DC\_Cmd, [242](#)
  - ADC\_DC\_DeInit, [242](#)
  - ADC\_DC\_GetLastData, [242](#)
  - ADC\_DC\_ITCmd, [243](#)
  - ADC\_DC\_ITConfig, [243](#)
  - ADC\_DC\_ITGenCmd, [244](#)
  - ADC\_DC\_ITMaskCmd, [244](#)
  - ADC\_DC\_ITMaskedStatus, [244](#)
  - ADC\_DC\_ITRawStatus, [245](#)
  - ADC\_DC\_ITStatusClear, [245](#)
  - ADC\_DC\_Init, [243](#)
  - ADC\_DC\_StructInit, [245](#)
  - ADC\_DC\_TrigStatus, [246](#)
  - ADC\_DC\_TrigStatusClear, [246](#)
  - ADC\_DeInit, [246](#)
  - ADC\_Init, [247](#)
  - ADC\_SEQ\_Cmd, [247](#)
  - ADC\_SEQ\_DMACmd, [248](#)
  - ADC\_SEQ\_DMAConfig, [248](#)
  - ADC\_SEQ\_DMAErrorStatus, [248](#)
  - ADC\_SEQ\_DMAErrorStatusClear, [249](#)
  - ADC\_SEQ\_DeInit, [247](#)
  - ADC\_SEQ\_FIFOEmptyStatus, [249](#)
  - ADC\_SEQ\_FIFOEmptyStatusClear, [249](#)
  - ADC\_SEQ\_FIFOFullStatus, [250](#)
  - ADC\_SEQ\_FIFOFullStatusClear, [250](#)
  - ADC\_SEQ\_GetConversionCount, [250](#)
  - ADC\_SEQ\_GetFIFOData, [251](#)
  - ADC\_SEQ\_GetFIFOLoad, [251](#)
  - ADC\_SEQ\_GetITCount, [251](#)
  - ADC\_SEQ\_ITCmd, [253](#)
  - ADC\_SEQ\_ITConfig, [253](#)
  - ADC\_SEQ\_ITCountRst, [253](#)
  - ADC\_SEQ\_ITMaskedStatus, [254](#)
  - ADC\_SEQ\_ITRawStatus, [254](#)
  - ADC\_SEQ\_ITStatusClear, [254](#)
  - ADC\_SEQ\_Init, [251](#)
  - ADC\_SEQ\_SWReq, [255](#)
  - ADC\_SEQ\_StructInit, [255](#)
  - ADC\_StructInit, [255](#)
  - BOOTFLASH\_FullErase, [259](#)
  - BOOTFLASH\_ITCmd, [260](#)
  - BOOTFLASH\_Info\_PageErase, [259](#)
  - BOOTFLASH\_Info\_Write, [260](#)
  - BOOTFLASH\_Init, [260](#)
  - BOOTFLASH\_OperationStatus, [260](#)
  - BOOTFLASH\_OperationStatusClear, [261](#)
  - BOOTFLASH\_PageErase, [261](#)
  - BOOTFLASH\_Write, [261](#)
  - CAP\_Capture\_Cmd, [266](#)
  - CAP\_Capture\_GetCap0, [266](#)
  - CAP\_Capture\_GetCap1, [267](#)
  - CAP\_Capture\_GetCap2, [267](#)
  - CAP\_Capture\_GetCap3, [267](#)
  - CAP\_Capture\_Init, [267](#)
  - CAP\_Capture\_SetCap0, [269](#)
  - CAP\_Capture\_SetCap1, [269](#)
  - CAP\_Capture\_SetCap2, [269](#)
  - CAP\_Capture\_SetCap3, [270](#)
  - CAP\_Capture\_StructInit, [270](#)
  - CAP\_DeInit, [270](#)
  - CAP\_GetShadowTimer, [271](#)
  - CAP\_GetTimer, [271](#)
  - CAP\_ITCmd, [271](#)
  - CAP\_ITForceCmd, [272](#)
  - CAP\_ITPendClear, [272](#)
  - CAP\_ITPendStatus, [272](#)
  - CAP\_ITStatus, [273](#)
  - CAP\_ITStatusClear, [273](#)
  - CAP\_Init, [271](#)
  - CAP\_PWM\_GetCompare, [273](#)
  - CAP\_PWM\_GetPeriod, [273](#)
  - CAP\_PWM\_GetShadowCompare, [274](#)
  - CAP\_PWM\_GetShadowPeriod, [274](#)
  - CAP\_PWM\_Init, [274](#)
  - CAP\_PWM\_SetCompare, [275](#)
  - 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\_AltFuncConfig, 306
- GPIO\_ClearBits, 307
- GPIO\_DeInit, 307
- GPIO\_ITCmd, 308
- GPIO\_ITConfig, 308
- GPIO\_ITStatusClear, 309
- GPIO\_Init, 307
- GPIO\_QualCmd, 309
- GPIO\_QualConfig, 309
- GPIO\_Read, 310
- GPIO\_ReadBit, 310
- GPIO\_ReadMask, 310
- GPIO\_SetBits, 311
- GPIO\_StructInit, 311
- GPIO\_SyncCmd, 311
- GPIO\_ToggleBits, 312
- GPIO\_Write, 312
- GPIO\_WriteBit, 312
- GPIO\_WriteMask, 313
- RCC\_ADCClkCmd, 319
- RCC\_ADCClkDivConfig, 319
- RCC\_PLLAutoConfig, 320
- RCC\_PLLDeInit, 321
- RCC\_PLLInit, 321
- RCC\_PLLPowerDownCmd, 322
- RCC\_PLLStructInit, 322
- RCC\_PeriphClkCmd, 319
- RCC\_PeriphRstCmd, 320
- RCC\_SPIClkCmd, 322
- RCC\_SPIClkDivConfig, 323
- RCC\_SPIClkSel, 323
- RCC\_SysClkDiv2Out, 323
- RCC\_SysClkSel, 324
- RCC\_SysClkStatus, 324
- RCC\_UARTClkCmd, 324
- RCC\_UARTClkDivConfig, 325
- RCC\_UARTClkSel, 325
- RCC\_USBClkCmd, 325
- RCC\_USBClkConfig, 326
- RCC\_WaitClkChange, 326
- TIMER\_Cmd, 333
- TIMER\_ExtInputConfig, 333
- TIMER\_FreqConfig, 334
- TIMER\_GetCounter, 334
- TIMER\_GetReload, 334
- TIMER\_ITCmd, 335
- TIMER\_ITStatus, 335
- TIMER\_ITStatusClear, 335
- TIMER\_PeriodConfig, 335
- TIMER\_SetCounter, 336
- TIMER\_SetReload, 336
- UART\_BaudRateDivConfig, 341
- UART\_Break, 341
- UART\_Cmd, 341
- UART\_DMABlkOnErrCmd, 343
- UART\_DMACmd, 343
- UART\_DeInit, 341
- UART\_ErrorStatus, 343
- UART\_ErrorStatusClear, 344
- UART\_FlagStatus, 344
- UART\_ITCmd, 345
- UART\_ITFIFOLevelConfig, 345
- UART\_ITMaskedStatus, 345
- UART\_ITRawStatus, 347
- UART\_ITStatusClear, 347
- UART\_Init, 344
- UART\_ModemConfig, 347
- UART\_ModemStructInit, 348
- UART\_RecieveData, 348
- UART\_SendData, 348
- UART\_StructInit, 349
- USERFLASH\_FullErase, 353
- USERFLASH\_ITCmd, 354
- USERFLASH\_Info\_PageErase, 353
- USERFLASH\_Info\_Read, 354
- USERFLASH\_Info\_Write, 354
- USERFLASH\_Init, 354
- USERFLASH\_OperationStatus, 355
- USERFLASH\_OperationStatusClear, 355
- USERFLASH\_PageErase, 355
- USERFLASH\_Read, 355
- USERFLASH\_Write, 356
- WATCHDOG\_Cmd, 360
- WATCHDOG\_GetCounter, 360
- WATCHDOG\_GetReload, 360
- WATCHDOG\_ITMaskedStatus, 361
- WATCHDOG\_ITRawStatus, 361
- WATCHDOG\_ITStatusClear, 361
- WATCHDOG\_LockCmd, 361
- WATCHDOG\_RstCmd, 361
- WATCHDOG\_SetReload, 362
- Приватные константы, 238, 258, 264, 281, 293, 299, 316, 332, 339, 352, 359
- WATCHDOG\_Lock\_Value, 359
- WATCHDOG\_Unlock\_Value, 359
- 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, [215](#)
  - [UART\\_ModemConfig, 215](#)
  - [UART\\_ModemStructInit, 215](#)
- 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, [124](#)
  - [DMA\\_ClearErrorStatus, 124](#)
  - [DMA\\_ErrorStatus, 124](#)
  - [DMA\\_MasterEnableStatus, 124](#)
  - [DMA\\_StateStatus, 124](#)
  - [DMA\\_WaitOnReqStatus, 125](#)
- Тактирование ADC, [182](#)
  - [RCC\\_ADCClkCmd, 182](#)
  - [RCC\\_ADCClkDivConfig, 182](#)
- Тактирование SPI, [180](#)
  - [RCC\\_SPIClkCmd, 180](#)
  - [RCC\\_SPIClkDivConfig, 180](#)
  - [RCC\\_SPIClkSel, 180](#)
- Тактирование UART, [177](#)
  - [RCC\\_UARTClkCmd, 177](#)
  - [RCC\\_UARTClkDivConfig, 177](#)
  - [RCC\\_UARTClkSel, 177](#)
- Тактирование USB, [176](#)
  - [RCC\\_USBClkCmd, 176](#)
  - [RCC\\_USBClkConfig, 176](#)
- Типы, [26, 65, 71, 110, 129, 135, 162, 184, 188, 197, 223, 230, 366](#)
  - [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, [138](#)
- Bit\_SET, [138](#)
- BitAction, [138](#)
- 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, [112](#)
- DMA\_ArbitrationRate\_1024, [113](#)
- DMA\_ArbitrationRate\_128, [113](#)
- DMA\_ArbitrationRate\_16, [112](#)
- DMA\_ArbitrationRate\_2, [112](#)
- DMA\_ArbitrationRate\_256, [113](#)
- DMA\_ArbitrationRate\_32, [113](#)
- DMA\_ArbitrationRate\_4, [112](#)
- DMA\_ArbitrationRate\_512, [113](#)
- DMA\_ArbitrationRate\_64, [113](#)
- DMA\_ArbitrationRate\_8, [112](#)
- DMA\_ArbitrationRate\_TypeDef, [112](#)
- DMA\_DataInc\_16, [113](#)
- DMA\_DataInc\_32, [113](#)
- DMA\_DataInc\_8, [113](#)
- DMA\_DataInc\_Disable, [113](#)
- DMA\_DataInc\_TypeDef, [113](#)
- DMA\_DataSize\_16, [113](#)
- DMA\_DataSize\_32, [113](#)
- DMA\_DataSize\_8, [113](#)
- DMA\_DataSize\_TypeDef, [113](#)
- DMA\_Mode\_AltMemScatGath, [113](#)
- DMA\_Mode\_AltPeriphScatGath, [113](#)
- DMA\_Mode\_AutoReq, [113](#)
- DMA\_Mode\_Basic, [113](#)
- DMA\_Mode\_Disable, [113](#)
- DMA\_Mode\_PingPong, [113](#)
- DMA\_Mode\_PrmMemScatGath, [113](#)
- DMA\_Mode\_PrmPeriphScatGath, [113](#)
- DMA\_Mode\_TypeDef, [113](#)
- DMA\_State\_Done, [114](#)
- DMA\_State\_Free, [114](#)
- DMA\_State\_Pause, [114](#)
- DMA\_State\_PeriphScatGath, [114](#)
- DMA\_State\_ReadConfigData, [114](#)
- DMA\_State\_ReadDstDataEndPtr, [114](#)

- DMA\_State\_ReadSrcData, [114](#)
- DMA\_State\_ReadSrcDataEndPtr, [114](#)
- DMA\_State\_TypeDef, [113](#)
- DMA\_State\_WaitReq, [114](#)
- DMA\_State\_WriteConfigData, [114](#)
- DMA\_State\_WriteDstData, [114](#)
- EXTMEM\_RWWaitState\_1, [131](#)
- EXTMEM\_RWWaitState\_2, [131](#)
- EXTMEM\_RWWaitState\_3, [131](#)
- EXTMEM\_RWWaitState\_4, [131](#)
- EXTMEM\_RWWaitState\_5, [131](#)
- EXTMEM\_RWWaitState\_6, [131](#)
- EXTMEM\_RWWaitState\_7, [131](#)
- EXTMEM\_RWWaitState\_8, [131](#)
- EXTMEM\_RWWaitState\_TypeDef, [131](#)
- EXTMEM\_ReadWaitState\_1, [131](#)
- EXTMEM\_ReadWaitState\_2, [131](#)
- EXTMEM\_ReadWaitState\_3, [131](#)
- EXTMEM\_ReadWaitState\_4, [131](#)
- EXTMEM\_ReadWaitState\_5, [131](#)
- EXTMEM\_ReadWaitState\_6, [131](#)
- EXTMEM\_ReadWaitState\_7, [131](#)
- EXTMEM\_ReadWaitState\_8, [131](#)
- EXTMEM\_ReadWaitState\_TypeDef, [131](#)
- EXTMEM\_Width\_16bit, [131](#)
- EXTMEM\_Width\_8bit, [131](#)
- EXTMEM\_Width\_TypeDef, [131](#)
- EXTMEM\_WriteWaitState\_1, [132](#)
- EXTMEM\_WriteWaitState\_2, [132](#)
- EXTMEM\_WriteWaitState\_3, [132](#)
- EXTMEM\_WriteWaitState\_4, [132](#)
- EXTMEM\_WriteWaitState\_5, [132](#)
- EXTMEM\_WriteWaitState\_6, [132](#)
- EXTMEM\_WriteWaitState\_7, [132](#)
- EXTMEM\_WriteWaitState\_8, [132](#)
- EXTMEM\_WriteWaitState\_TypeDef, [131](#)
- GPIO\_AltFunc\_1, [139](#)
- GPIO\_AltFunc\_2, [139](#)
- GPIO\_AltFunc\_3, [139](#)
- GPIO\_AltFunc\_TypeDef, [138](#)
- GPIO\_Dir\_In, [139](#)
- GPIO\_Dir\_Out, [139](#)
- GPIO\_Dir\_TypeDef, [139](#)
- GPIO\_IntPol\_Neg, [139](#)
- GPIO\_IntPol\_Pos, [139](#)
- GPIO\_IntPol\_TypeDef, [139](#)
- GPIO\_IntType\_Edge, [139](#)
- GPIO\_IntType\_Level, [139](#)
- GPIO\_IntType\_TypeDef, [139](#)
- GPIO\_Load\_16mA, [139](#)
- GPIO\_Load\_8mA, [139](#)
- GPIO\_Load\_TypeDef, [139](#)
- GPIO\_Mode\_AltFunc, [140](#)
- GPIO\_Mode\_IO, [140](#)
- GPIO\_Mode\_TypeDef, [139](#)
- GPIO\_Out\_Dis, [140](#)
- GPIO\_Out\_En, [140](#)
- GPIO\_Out\_TypeDef, [140](#)
- GPIO\_OutMode\_OD, [140](#)
- GPIO\_OutMode\_PP, [140](#)
- GPIO\_OutMode\_TypeDef, [140](#)
- GPIO\_PullUp\_Dis, [140](#)
- GPIO\_PullUp\_En, [140](#)
- GPIO\_PullUp\_TypeDef, [140](#)
- GPIO\_Qual\_Dis, [140](#)
- GPIO\_Qual\_En, [140](#)
- GPIO\_Qual\_TypeDef, [140](#)
- GPIO\_QualMode\_3sample, [141](#)
- GPIO\_QualMode\_6sample, [141](#)
- GPIO\_QualMode\_TypeDef, [140](#)
- GPIO\_Sync\_Dis, [141](#)
- GPIO\_Sync\_En, [141](#)
- GPIO\_Sync\_TypeDef, [141](#)
- 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, [366](#)
- 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, [111](#)
- IS\_DMA\_DATA\_INC, [111](#)
- IS\_DMA\_DATA\_SIZE, [111](#)
- IS\_DMA\_MODE, [112](#)
- IS\_DMA\_STATE, [112](#)
- IS\_EXTMEM\_READ\_WAITSTATE, [129](#)
- IS\_EXTMEM\_RW\_WAITSTATE, [130](#)
- IS\_EXTMEM\_WIDTH, [130](#)
- IS\_EXTMEM\_WRITE\_WAITSTATE, [130](#)
- IS\_GPIO\_ALL\_PERIPH, [366](#)
- IS\_GPIO\_ALT\_FUNC, [136](#)
- IS\_GPIO\_DIR, [136](#)
- IS\_GPIO\_INT\_POL, [136](#)
- IS\_GPIO\_INT\_TYPE, [136](#)
- IS\_GPIO\_LOAD, [137](#)
- IS\_GPIO\_MODE, [137](#)
- IS\_GPIO\_OUT, [137](#)
- IS\_GPIO\_OUT\_MODE, [137](#)
- IS\_GPIO\_PULLUP, [137](#)
- IS\_GPIO\_QUAL, [138](#)
- IS\_GPIO\_QUAL\_MODE, [138](#)
- IS\_GPIO\_SYNC, [138](#)
- IS\_RCC\_ADC\_CLK, [164](#)

- IS\_RCC\_PERIPH\_CLK, 164
- IS\_RCC\_PLL\_NO, 164
- IS\_RCC\_PLL\_REF, 164
- IS\_RCC\_SPI\_CLK, 165
- IS\_RCC\_SYS\_CLK, 165
- IS\_RCC\_UART\_CLK, 165
- IS\_RCC\_USB\_CLK, 165
- IS\_RCC\_USB\_FREQ, 166
- IS\_RTC\_FORMAT, 185
- IS\_RTC\_MONTH, 185
- IS\_RTC\_WEEKDAY, 185
- IS\_SPI\_ALL\_PERIPH, 367
- IS\_TIMER\_ALL\_PERIPH, 367
- IS\_TIMER\_EXT\_INPUT, 188
- IS\_UART\_ALL\_PERIPH, 367
- IS\_UART\_DATA\_WIDTH, 198
- IS\_UART\_DIR, 198
- IS\_UART\_FIFO\_LEVEL, 198
- IS\_UART\_PARITY\_BIT, 198
- IS\_UART\_STOP\_BIT, 198
- IS\_USERFLASH\_STATUS, 223
- RCC\_ADCClk\_0, 166
- RCC\_ADCClk\_1, 166
- RCC\_ADCClk\_10, 166
- RCC\_ADCClk\_11, 166
- RCC\_ADCClk\_2, 166
- RCC\_ADCClk\_3, 166
- RCC\_ADCClk\_4, 166
- RCC\_ADCClk\_5, 166
- RCC\_ADCClk\_6, 166
- RCC\_ADCClk\_7, 166
- RCC\_ADCClk\_8, 166
- RCC\_ADCClk\_9, 166
- RCC\_ADCClk\_TypeDef, 166
- RCC\_PLLNO\_Disable, 168
- RCC\_PLLNO\_Div2, 168
- RCC\_PLLNO\_Div4, 168
- RCC\_PLLNO\_TypeDef, 168
- RCC\_PLLRef\_ETH\_25MHz, 168
- RCC\_PLLRef\_TypeDef, 168
- RCC\_PLLRef\_USB\_60MHz, 168
- RCC\_PLLRef\_USB\_CLK, 168
- RCC\_PLLRef\_XI\_OSC, 168
- RCC\_PeriphClk\_ADC, 167
- RCC\_PeriphClk\_CMP, 166
- RCC\_PeriphClk\_I2C0, 167
- RCC\_PeriphClk\_I2C1, 167
- RCC\_PeriphClk\_PWM0, 166
- RCC\_PeriphClk\_PWM1, 166
- RCC\_PeriphClk\_PWM2, 166
- RCC\_PeriphClk\_PWM3, 166
- RCC\_PeriphClk\_PWM4, 167
- RCC\_PeriphClk\_PWM5, 167
- RCC\_PeriphClk\_PWM6, 167
- RCC\_PeriphClk\_PWM7, 167
- RCC\_PeriphClk\_PWM8, 167
- RCC\_PeriphClk\_QEP0, 166
- RCC\_PeriphClk\_QEP1, 166
- RCC\_PeriphClk\_TypeDef, 166
- RCC\_PeriphClk\_WD, 167
- RCC\_PeriphRst\_CAP0, 167
- RCC\_PeriphRst\_CAP1, 167
- RCC\_PeriphRst\_CAP2, 168
- RCC\_PeriphRst\_CAP3, 168
- RCC\_PeriphRst\_CAP4, 168
- RCC\_PeriphRst\_CAP5, 168
- RCC\_PeriphRst\_CMP, 168
- RCC\_PeriphRst\_ETH, 167
- RCC\_PeriphRst\_I2C0, 167
- RCC\_PeriphRst\_I2C1, 167
- RCC\_PeriphRst\_PWM0, 167
- RCC\_PeriphRst\_PWM1, 167
- RCC\_PeriphRst\_PWM2, 167
- RCC\_PeriphRst\_PWM3, 167
- RCC\_PeriphRst\_PWM4, 167
- RCC\_PeriphRst\_PWM5, 167
- RCC\_PeriphRst\_PWM6, 167
- RCC\_PeriphRst\_PWM7, 167
- RCC\_PeriphRst\_PWM8, 167
- RCC\_PeriphRst\_QEP0, 167
- RCC\_PeriphRst\_QEP1, 167
- RCC\_PeriphRst\_SPI0, 167
- RCC\_PeriphRst\_SPI1, 167
- RCC\_PeriphRst\_SPI2, 167
- RCC\_PeriphRst\_SPI3, 167
- RCC\_PeriphRst\_Timer0, 167
- RCC\_PeriphRst\_Timer1, 167
- RCC\_PeriphRst\_Timer2, 167
- RCC\_PeriphRst\_TypeDef, 167
- RCC\_PeriphRst\_UART0, 167
- RCC\_PeriphRst\_UART1, 167
- RCC\_PeriphRst\_UART2, 167
- RCC\_PeriphRst\_UART3, 167
- RCC\_PeriphRst\_USB, 167
- RCC\_PeriphRst\_WD, 167
- RCC\_SPIClk\_SYSCLK, 168
- RCC\_SPIClk\_TypeDef, 168
- RCC\_SPIClk\_USB\_60MHz, 168
- RCC\_SPIClk\_USB\_CLK, 168
- RCC\_SPIClk\_XI\_OSC, 168
- RCC\_SysClk\_CPE\_Sel, 169
- RCC\_SysClk\_ETH25MHz, 169
- RCC\_SysClk\_PLL, 169
- RCC\_SysClk\_PLLDIV, 169
- RCC\_SysClk\_POR, 169
- RCC\_SysClk\_TypeDef, 168
- RCC\_SysClk\_USB60MHz, 169
- RCC\_SysClk\_USB\_CLK, 169
- RCC\_SysClk\_XI\_OSC, 169
- RCC\_UARTClk\_SYSCLK, 169
- RCC\_UARTClk\_TypeDef, 169
- RCC\_UARTClk\_USB\_60MHz, 169
- RCC\_UARTClk\_USB\_CLK, 169
- RCC\_UARTClk\_XI\_OSC, 169
- RCC\_USBClk\_TypeDef, 169
- RCC\_USBClk\_USB\_CLK, 169

- RCC\_USBCLK\_XI\_OSC, 169
- RCC\_USBFreq\_12MHz, 169
- RCC\_USBFreq\_24MHz, 169
- RCC\_USBFreq\_TypeDef, 169
- RTC\_Format\_BCD, 185
- RTC\_Format\_BIN, 185
- RTC\_Format\_TypeDef, 185
- RTC\_Month\_April, 186
- RTC\_Month\_August, 186
- RTC\_Month\_December, 186
- RTC\_Month\_February, 186
- RTC\_Month\_January, 186
- RTC\_Month\_July, 186
- RTC\_Month\_June, 186
- RTC\_Month\_March, 186
- RTC\_Month\_May, 186
- RTC\_Month\_November, 186
- RTC\_Month\_October, 186
- RTC\_Month\_September, 186
- RTC\_Month\_TypeDef, 185
- RTC\_Weekday\_Friday, 186
- RTC\_Weekday\_Monday, 186
- RTC\_Weekday\_Saturday, 186
- RTC\_Weekday\_Sunday, 186
- RTC\_Weekday\_Thursday, 186
- RTC\_Weekday\_Tuesday, 186
- RTC\_Weekday\_TypeDef, 186
- RTC\_Weekday\_Wednesday, 186
- TIMER\_ExtInput\_CountClk, 188
- TIMER\_ExtInput\_CountEn, 188
- TIMER\_ExtInput\_Disable, 188
- TIMER\_ExtInput\_TypeDef, 188
- UART\_DataWidth\_5, 199
- UART\_DataWidth\_6, 199
- UART\_DataWidth\_7, 199
- UART\_DataWidth\_8, 199
- UART\_DataWidth\_TypeDef, 199
- UART\_Dir\_Rx, 199
- UART\_Dir\_Tx, 199
- UART\_Dir\_Typedef, 199
- UART\_FIFOLevel\_1\_2, 199
- UART\_FIFOLevel\_1\_4, 199
- UART\_FIFOLevel\_1\_8, 199
- UART\_FIFOLevel\_3\_4, 199
- UART\_FIFOLevel\_7\_8, 199
- UART\_FIFOLevel\_TypeDef, 199
- UART\_ParityBit\_Disable, 200
- UART\_ParityBit\_Even, 200
- UART\_ParityBit\_High, 200
- UART\_ParityBit\_Low, 200
- UART\_ParityBit\_Odd, 200
- UART\_ParityBit\_TypeDef, 199
- UART\_StopBit\_1, 200
- UART\_StopBit\_2, 200
- UART\_StopBit\_TypeDef, 200
- USERFLASH\_Status\_Complete, 223
- USERFLASH\_Status\_Error, 223
- USERFLASH\_Status\_None, 223
- USERFLASH\_Status\_TypeDef, 223
- Управление сбросом, 183
  - RCC\_PeriphRstCmd, 183
- Управление тактированием, 174
  - RCC\_PeriphClkCmd, 174
  - RCC\_SysClkSel, 174
  - RCC\_SysClkStatus, 175
- Запись, 153
  - GPIO\_Write, 153
  - GPIO\_WriteBit, 153
  - GPIO\_WriteMask, 153
- ADC, 236
- ADC\_Average
  - ADC\_Init\_TypeDef, 375
- ADC\_Average\_16
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_2
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_32
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_4
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_64
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_8
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_Disable
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Average\_TypeDef
  - Типы, 32
  - niietcm4\_adc.h, 437
- ADC\_Channel\_0
  - Маски каналов для измерений, 16
  - niietcm4\_adc.h, 427
- ADC\_Channel\_1
  - Маски каналов для измерений, 16
  - niietcm4\_adc.h, 427
- ADC\_Channel\_10
  - Маски каналов для измерений, 16
  - niietcm4\_adc.h, 427
- ADC\_Channel\_11
  - Маски каналов для измерений, 16
  - niietcm4\_adc.h, 427
- ADC\_Channel\_12
  - Маски каналов для измерений, 17
  - niietcm4\_adc.h, 427
- ADC\_Channel\_13
  - Маски каналов для измерений, 17
  - niietcm4\_adc.h, 427
- ADC\_Channel\_14
  - Маски каналов для измерений, 17



- niietcm4\_adc.h, [427](#)
- ADC\_Channel\_15
  - Маски каналов для измерений, [17](#)
  - niietcm4\_adc.h, [427](#)
- ADC\_Channel\_16
  - Маски каналов для измерений, [17](#)
  - niietcm4\_adc.h, [427](#)
- ADC\_Channel\_17
  - Маски каналов для измерений, [17](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_18
  - Маски каналов для измерений, [17](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_19
  - Маски каналов для измерений, [17](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_2
  - Маски каналов для измерений, [17](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_20
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_21
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_22
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_23
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_3
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [428](#)
- ADC\_Channel\_4
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_5
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_6
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_7
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_8
  - Маски каналов для измерений, [18](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_9
  - Маски каналов для измерений, [19](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_All
  - Маски каналов для измерений, [19](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channel\_None
  - Маски каналов для измерений, [19](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_Channels
  - ADC\_SEQ\_Init\_TypeDef, [376](#)
- ADC\_Cmd
  - Функции, [37](#)
  - Приватные функции, [242](#)
  - niietcm4\_adc.c, [405](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_DC\_0
  - Маски выбора цифровых компараторов, [20](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_DC\_1
  - Маски выбора цифровых компараторов, [20](#)
  - niietcm4\_adc.h, [429](#)
- ADC\_DC\_10
  - Маски выбора цифровых компараторов, [20](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_11
  - Маски выбора цифровых компараторов, [20](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_12
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_13
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_14
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_15
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_16
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_17
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_18
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [430](#)
- ADC\_DC\_19
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [431](#)
- ADC\_DC\_2
  - Маски выбора цифровых компараторов, [21](#)
  - niietcm4\_adc.h, [431](#)
- ADC\_DC\_20
  - Маски выбора цифровых компараторов, [22](#)
  - niietcm4\_adc.h, [431](#)
- ADC\_DC\_21
  - Маски выбора цифровых компараторов, [22](#)
  - niietcm4\_adc.h, [431](#)
- ADC\_DC\_22
  - Маски выбора цифровых компараторов, [22](#)
  - niietcm4\_adc.h, [431](#)
- ADC\_DC\_23
  - Маски выбора цифровых компараторов, [22](#)
  - niietcm4\_adc.h, [431](#)

- ADC\_DC\_3
  - Маски выбора цифровых компараторов, [22](#)
  - [niietcm4\\_adc.h](#), [431](#)
- ADC\_DC\_4
  - Маски выбора цифровых компараторов, [22](#)
  - [niietcm4\\_adc.h](#), [431](#)
- ADC\_DC\_5
  - Маски выбора цифровых компараторов, [22](#)
  - [niietcm4\\_adc.h](#), [431](#)
- ADC\_DC\_6
  - Маски выбора цифровых компараторов, [22](#)
  - [niietcm4\\_adc.h](#), [431](#)
- ADC\_DC\_7
  - Маски выбора цифровых компараторов, [22](#)
  - [niietcm4\\_adc.h](#), [432](#)
- ADC\_DC\_8
  - Маски выбора цифровых компараторов, [22](#)
  - [niietcm4\\_adc.h](#), [432](#)
- ADC\_DC\_9
  - Маски выбора цифровых компараторов, [23](#)
  - [niietcm4\\_adc.h](#), [432](#)
- ADC\_DC\_All
  - Маски выбора цифровых компараторов, [23](#)
  - [niietcm4\\_adc.h](#), [432](#)
- ADC\_DC\_Channel
  - ADC\_DC\_Init\_TypeDef, [374](#)
- ADC\_DC\_Channel\_0
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_1
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_10
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_11
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_12
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_13
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_14
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_15
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_16
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_17
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_18
  - Типы, [33](#)
- [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_19
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_2
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_20
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_21
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_22
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_23
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_3
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_4
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_5
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_6
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_7
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_8
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Channel\_9
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_None
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Channel\_TypeDef
  - Типы, [32](#)
  - [niietcm4\\_adc.h](#), [437](#)
- ADC\_DC\_Cmd
  - Функции, [38](#)
  - Приватные функции, [242](#)
  - [niietcm4\\_adc.c](#), [406](#)
  - [niietcm4\\_adc.h](#), [442](#)
- ADC\_DC\_Condition
  - ADC\_DC\_Init\_TypeDef, [374](#)
- ADC\_DC\_Condition\_High
  - Типы, [33](#)
  - [niietcm4\\_adc.h](#), [438](#)
- ADC\_DC\_Condition\_Low
  - Типы, [33](#)

- niietcm4\_adc.h, 438
- ADC\_DC\_Condition\_TypeDef
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_Condition\_Window
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_DeInit
  - Цифровые компараторы, 46
  - Приватные функции, 242
- niietcm4\_adc.c, 406
  - niietcm4\_adc.h, 442
- ADC\_DC\_GetLastData
  - Функции, 38
  - Приватные функции, 242
- niietcm4\_adc.c, 406
  - niietcm4\_adc.h, 442
- ADC\_DC\_ITCmd
  - Цифровые компараторы, 55
  - Приватные функции, 243
- niietcm4\_adc.c, 407
  - niietcm4\_adc.h, 444
- ADC\_DC\_ITConfig
  - Цифровые компараторы, 55
  - Приватные функции, 243
- niietcm4\_adc.c, 407
  - niietcm4\_adc.h, 444
- ADC\_DC\_ITGenCmd
  - Цифровые компараторы, 57
  - Приватные функции, 244
- niietcm4\_adc.c, 408
  - niietcm4\_adc.h, 444
- ADC\_DC\_ITMaskCmd
  - Цифровые компараторы, 57
  - Приватные функции, 244
- niietcm4\_adc.c, 408
  - niietcm4\_adc.h, 446
- ADC\_DC\_ITMaskedStatus
  - Цифровые компараторы, 57
  - Приватные функции, 244
- niietcm4\_adc.c, 408
  - niietcm4\_adc.h, 446
- ADC\_DC\_ITRawStatus
  - Цифровые компараторы, 58
  - Приватные функции, 245
- niietcm4\_adc.c, 409
  - niietcm4\_adc.h, 446
- ADC\_DC\_ITStatusClear
  - Цифровые компараторы, 58
  - Приватные функции, 245
- niietcm4\_adc.c, 409
  - niietcm4\_adc.h, 447
- ADC\_DC\_Init
  - Цифровые компараторы, 46
  - Приватные функции, 243
- niietcm4\_adc.c, 407
  - niietcm4\_adc.h, 442
- ADC\_DC\_Init\_TypeDef, 373
  - ADC\_DC\_Channel, 374
  - ADC\_DC\_Condition, 374
  - ADC\_DC\_Mode, 374
  - ADC\_DC\_ThresholdHigh, 374
  - ADC\_DC\_ThresholdLow, 374
- ADC\_DC\_Mode
  - ADC\_DC\_Init\_TypeDef, 374
- ADC\_DC\_Mode\_Multiple
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_Mode\_MultipleHyst
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_Mode\_Single
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_Mode\_SingleHyst
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_Mode\_TypeDef
  - Типы, 33
- niietcm4\_adc.h, 438
- ADC\_DC\_Module\_0
  - Типы, 33
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_1
  - Типы, 33
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_10
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_11
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_12
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_13
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_14
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_15
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_16
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_17
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_18
  - Типы, 34
- niietcm4\_adc.h, 439
- ADC\_DC\_Module\_19
  - Типы, 34
- niietcm4\_adc.h, 439



- ADC\_DC\_Module\_2
  - Типы, [33](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_20
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_21
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_22
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_23
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_3
  - Типы, [33](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_4
  - Типы, [33](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_5
  - Типы, [33](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_6
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_7
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_8
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_9
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_DC\_Module\_TypeDef
  - Типы, [33](#)
  - niietcm4\_adc.h, [438](#)
- ADC\_DC\_None
  - Маски выбора цифровых компараторов, [23](#)
  - niietcm4\_adc.h, [432](#)
- ADC\_DC\_StructInit
  - Цифровые компараторы, [46](#)
  - Приватные функции, [245](#)
  - niietcm4\_adc.c, [409](#)
  - niietcm4\_adc.h, [447](#)
- ADC\_DC\_ThresholdHigh
  - ADC\_DC\_Init\_TypeDef, [374](#)
- ADC\_DC\_ThresholdLow
  - ADC\_DC\_Init\_TypeDef, [374](#)
- ADC\_DC\_TrigStatus
  - Функции, [38](#)
  - Приватные функции, [246](#)
  - niietcm4\_adc.c, [410](#)
  - niietcm4\_adc.h, [447](#)
- ADC\_DC\_TrigStatusClear
  - Функции, [39](#)
- Приватные функции, [246](#)
- niietcm4\_adc.c, [410](#)
- niietcm4\_adc.h, [447](#)
- ADC\_DeInit
  - Модули АЦП, [44](#)
  - Приватные функции, [246](#)
  - niietcm4\_adc.c, [410](#)
  - niietcm4\_adc.h, [448](#)
- ADC\_Init
  - Модули АЦП, [44](#)
  - Приватные функции, [247](#)
  - niietcm4\_adc.c, [410](#)
  - niietcm4\_adc.h, [448](#)
- ADC\_Init\_TypeDef, [375](#)
  - ADC\_Average, [375](#)
  - ADC\_Measure\_A, [375](#)
  - ADC\_Measure\_B, [375](#)
  - ADC\_Mode, [375](#)
  - ADC\_Phase, [376](#)
  - ADC\_Resolution, [376](#)
- ADC\_Measure\_A
  - ADC\_Init\_TypeDef, [375](#)
- ADC\_Measure\_B
  - ADC\_Init\_TypeDef, [375](#)
- ADC\_Measure\_Diff
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Measure\_Single
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Measure\_TypeDef
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Mode
  - ADC\_Init\_TypeDef, [375](#)
- ADC\_Mode\_Active
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Mode\_Powerdown
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Mode\_StandBy
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Mode\_TypeDef
  - Типы, [34](#)
  - niietcm4\_adc.h, [439](#)
- ADC\_Module\_0
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_Module\_1
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_Module\_10
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_Module\_11
  - Типы, [35](#)

- niietcm4\_adc.h, 440
- ADC\_Module\_2
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_3
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_4
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_5
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_6
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_7
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_8
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_9
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Module\_TypeDef
  - Типы, 34
  - niietcm4\_adc.h, 439
- ADC\_Phase
  - ADC\_Init\_TypeDef, 376
- ADC\_Resolution
  - ADC\_Init\_TypeDef, 376
- ADC\_Resolution\_10bit
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Resolution\_12bit
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_Resolution\_TypeDef
  - Типы, 35
  - niietcm4\_adc.h, 440
- ADC\_SEQ\_0
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 432
- ADC\_SEQ\_1
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 432
- ADC\_SEQ\_2
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 432
- ADC\_SEQ\_3
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 432
- ADC\_SEQ\_4
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 433
- ADC\_SEQ\_5
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 433
- ADC\_SEQ\_6
  - Маски выбора секвенсоров, 24
  - niietcm4\_adc.h, 433
- ADC\_SEQ\_7
  - Маски выбора секвенсоров, 25
  - niietcm4\_adc.h, 433
- ADC\_SEQ\_Cmd
  - Функции, 39
  - Приватные функции, 247
  - niietcm4\_adc.c, 412
  - niietcm4\_adc.h, 448
- ADC\_SEQ\_ConversionCount
  - ADC\_SEQ\_Init\_TypeDef, 377
- ADC\_SEQ\_ConversionDelay
  - ADC\_SEQ\_Init\_TypeDef, 377
- ADC\_SEQ\_DC
  - ADC\_SEQ\_Init\_TypeDef, 377
- ADC\_SEQ\_DMAMcmd
  - Конфигурация секвенсоров для DMA, 51
  - Приватные функции, 248
  - niietcm4\_adc.c, 412
  - niietcm4\_adc.h, 449
- ADC\_SEQ\_DMAConfig
  - Конфигурация секвенсоров для DMA, 51
  - Приватные функции, 248
  - niietcm4\_adc.c, 413
  - niietcm4\_adc.h, 449
- ADC\_SEQ\_DMAErrorStatus
  - Конфигурация секвенсоров для DMA, 51
  - Приватные функции, 248
  - niietcm4\_adc.c, 413
  - niietcm4\_adc.h, 450
- ADC\_SEQ\_DMAErrorStatusClear
  - Конфигурация секвенсоров для DMA, 53
  - Приватные функции, 249
  - niietcm4\_adc.c, 413
  - niietcm4\_adc.h, 450
- ADC\_SEQ\_DeInit
  - Приватные функции, 247
  - Секвенсоры, 48
  - niietcm4\_adc.c, 412
  - niietcm4\_adc.h, 449
- ADC\_SEQ\_FIFOEmptyStatus
  - Функции, 39
  - Приватные функции, 249
  - niietcm4\_adc.c, 414
  - niietcm4\_adc.h, 450
- ADC\_SEQ\_FIFOEmptyStatusClear
  - Функции, 40
  - Приватные функции, 249
  - niietcm4\_adc.c, 414
  - niietcm4\_adc.h, 451
- ADC\_SEQ\_FIFOFullStatus
  - Функции, 40
  - Приватные функции, 250
  - niietcm4\_adc.c, 414
  - niietcm4\_adc.h, 451

- ADC\_SEQ\_FIFOFullStatusClear
  - Функции, [40](#)
  - Приватные функции, [250](#)
  - niietcm4\_adc.c, [415](#)
  - niietcm4\_adc.h, [451](#)
- ADC\_SEQ\_FIFOLevel\_1
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_FIFOLevel\_16
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_FIFOLevel\_2
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_FIFOLevel\_32
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_FIFOLevel\_4
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_FIFOLevel\_8
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_FIFOLevel\_TypeDef
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_GetConversionCount
  - Функции, [41](#)
  - Приватные функции, [250](#)
  - niietcm4\_adc.c, [415](#)
  - niietcm4\_adc.h, [451](#)
- ADC\_SEQ\_GetFIFOData
  - Функции, [41](#)
  - Приватные функции, [251](#)
  - niietcm4\_adc.c, [415](#)
  - niietcm4\_adc.h, [453](#)
- ADC\_SEQ\_GetFIFOLoad
  - Функции, [41](#)
  - Приватные функции, [251](#)
  - niietcm4\_adc.c, [415](#)
  - niietcm4\_adc.h, [453](#)
- ADC\_SEQ\_GetITCount
  - Приватные функции, [251](#)
  - Секвенсоры, [59](#)
  - niietcm4\_adc.c, [417](#)
  - niietcm4\_adc.h, [453](#)
- ADC\_SEQ\_ITCmd
  - Приватные функции, [253](#)
  - Секвенсоры, [59](#)
  - niietcm4\_adc.c, [417](#)
  - niietcm4\_adc.h, [454](#)
- ADC\_SEQ\_ITConfig
  - Приватные функции, [253](#)
  - Секвенсоры, [60](#)
  - niietcm4\_adc.c, [418](#)
  - niietcm4\_adc.h, [454](#)
- ADC\_SEQ\_ITCountRst
  - Приватные функции, [253](#)
- Секвенсоры, [60](#)
- niietcm4\_adc.c, [418](#)
- niietcm4\_adc.h, [455](#)
- ADC\_SEQ\_ITMaskedStatus
  - Приватные функции, [254](#)
  - Секвенсоры, [60](#)
  - niietcm4\_adc.c, [418](#)
  - niietcm4\_adc.h, [455](#)
- ADC\_SEQ\_ITRawStatus
  - Приватные функции, [254](#)
  - Секвенсоры, [61](#)
  - niietcm4\_adc.c, [419](#)
  - niietcm4\_adc.h, [455](#)
- ADC\_SEQ\_ITStatusClear
  - Приватные функции, [254](#)
  - Секвенсоры, [61](#)
  - niietcm4\_adc.c, [419](#)
  - niietcm4\_adc.h, [456](#)
- ADC\_SEQ\_Init
  - Приватные функции, [251](#)
  - Секвенсоры, [48](#)
  - niietcm4\_adc.c, [417](#)
  - niietcm4\_adc.h, [454](#)
- ADC\_SEQ\_Init\_TypeDef, [376](#)
  - ADC\_Channels, [376](#)
  - ADC\_SEQ\_ConversionCount, [377](#)
  - ADC\_SEQ\_ConversionDelay, [377](#)
  - ADC\_SEQ\_DC, [377](#)
  - ADC\_SEQ\_SWReqEn, [377](#)
  - ADC\_SEQ\_StartEvent, [377](#)
- ADC\_SEQ\_Module\_0
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_1
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_2
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_3
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_4
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_5
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_6
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_7
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_Module\_TypeDef
  - Типы, [35](#)
  - niietcm4\_adc.h, [440](#)
- ADC\_SEQ\_SWReq

- Функции, [41](#)
- Приватные функции, [255](#)
- niietcm4\_adc.c, [420](#)
- niietcm4\_adc.h, [456](#)
- ADC\_SEQ\_SWReqEn
  - ADC\_SEQ\_Init\_TypeDef, [377](#)
- ADC\_SEQ\_StartEvent
  - ADC\_SEQ\_Init\_TypeDef, [377](#)
- ADC\_SEQ\_StartEvent\_CMP0
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_CMP1
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_CMP2
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_Cycle
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_ITGPIO
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_PWM0
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_PWM1
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_PWM2
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_PWM3
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_PWM4
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_PWM5
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_SWReq
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_TIM
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StartEvent\_TypeDef
  - Типы, [36](#)
  - niietcm4\_adc.h, [441](#)
- ADC\_SEQ\_StructInit
  - Приватные функции, [255](#)
  - Секвенсоры, [48](#)
  - niietcm4\_adc.c, [419](#)
  - niietcm4\_adc.h, [456](#)
- ADC\_StructInit
  - Модули АЦП, [44](#)
  - Приватные функции, [255](#)
- niietcm4\_adc.c, [420](#)
- niietcm4\_adc.h, [456](#)
- ALT\_DATA
  - DMA\_ConfigData\_TypeDef, [386](#)
- BOOTFLASH, [256](#)
- BOOTFLASH\_FullErase
  - Основная область флеш, [68](#)
  - Приватные функции, [259](#)
  - niietcm4\_bootflash.c, [459](#)
  - niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_INFO\_PAGE\_SIZE\_BYTES
  - Информационная область флеш, [64](#)
  - niietcm4\_bootflash.h, [463](#)
- BOOTFLASH\_INFO\_PAGE\_TOTAL
  - Информационная область флеш, [64](#)
  - niietcm4\_bootflash.h, [463](#)
- BOOTFLASH\_INFO\_TOTAL\_BYTES
  - Информационная область флеш, [64](#)
  - niietcm4\_bootflash.h, [463](#)
- BOOTFLASH\_ITCmd
  - Функции, [66](#)
  - Приватные функции, [260](#)
  - niietcm4\_bootflash.c, [460](#)
  - niietcm4\_bootflash.h, [465](#)
- BOOTFLASH\_Info\_PageErase
  - Информационная область флеш, [70](#)
  - Приватные функции, [259](#)
  - niietcm4\_bootflash.c, [459](#)
  - niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_Info\_Write
  - Информационная область флеш, [70](#)
  - Приватные функции, [260](#)
  - niietcm4\_bootflash.c, [459](#)
  - niietcm4\_bootflash.h, [465](#)
- BOOTFLASH\_Init
  - Функции, [66](#)
  - Приватные функции, [260](#)
  - niietcm4\_bootflash.c, [460](#)
  - niietcm4\_bootflash.h, [465](#)
- BOOTFLASH\_OperationStatus
  - Функции, [66](#)
  - Приватные функции, [260](#)
  - niietcm4\_bootflash.c, [460](#)
  - niietcm4\_bootflash.h, [465](#)
- BOOTFLASH\_OperationStatusClear
  - Функции, [67](#)
  - Приватные функции, [261](#)
  - niietcm4\_bootflash.c, [460](#)
  - niietcm4\_bootflash.h, [466](#)
- BOOTFLASH\_PAGE\_SIZE\_BYTES
  - Основная область флеш, [63](#)
  - niietcm4\_bootflash.h, [463](#)
- BOOTFLASH\_PAGE\_TOTAL
  - Основная область флеш, [63](#)
  - niietcm4\_bootflash.h, [463](#)
- BOOTFLASH\_PageErase
  - Основная область флеш, [68](#)
  - Приватные функции, [261](#)

- niietcm4\_bootflash.c, [461](#)
  - niietcm4\_bootflash.h, [466](#)
- BOOTFLASH\_Status\_Complete
  - Типы, [65](#)
- niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_Status\_Error
  - Типы, [65](#)
- niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_Status\_None
  - Типы, [65](#)
- niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_Status\_TypeDef
  - Типы, [65](#)
- niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_TOTAL\_BYTES
  - Основная область флеш, [63](#)
- niietcm4\_bootflash.h, [464](#)
- BOOTFLASH\_Write
  - Основная область флеш, [68](#)
  - Приватные функции, [261](#)
- niietcm4\_bootflash.c, [461](#)
  - niietcm4\_bootflash.h, [466](#)
- BUFFERABLE
  - DMA\_Protect\_TypeDef, [388](#)
- Bit\_CLEAR
  - Типы, [138](#)
- niietcm4\_gpio.h, [560](#)
- Bit\_SET
  - Типы, [138](#)
- niietcm4\_gpio.h, [560](#)
- BitAction
  - Типы, [138](#)
- niietcm4\_gpio.h, [560](#)
- CACHEABLE
  - DMA\_Protect\_TypeDef, [388](#)
- CAP, [262](#)
- CAP\_Capture\_Cmd
  - Приватные функции, [266](#)
  - Режим захвата, [89](#)
- niietcm4\_cap.c, [469](#)
  - niietcm4\_cap.h, [487](#)
- CAP\_Capture\_GetCap0
  - Приватные функции, [266](#)
  - Режим захвата, [89](#)
- niietcm4\_cap.c, [469](#)
  - niietcm4\_cap.h, [488](#)
- CAP\_Capture\_GetCap1
  - Приватные функции, [267](#)
  - Режим захвата, [90](#)
- niietcm4\_cap.c, [469](#)
  - niietcm4\_cap.h, [488](#)
- CAP\_Capture\_GetCap2
  - Приватные функции, [267](#)
  - Режим захвата, [90](#)
- niietcm4\_cap.c, [470](#)
  - niietcm4\_cap.h, [488](#)
- CAP\_Capture\_GetCap3
  - Приватные функции, [267](#)
- Режим захвата, [90](#)
- niietcm4\_cap.c, [470](#)
  - niietcm4\_cap.h, [488](#)
- CAP\_Capture\_Init
  - Приватные функции, [267](#)
  - Режим захвата, [90](#)
- niietcm4\_cap.c, [470](#)
  - niietcm4\_cap.h, [489](#)
- CAP\_Capture\_Init\_TypeDef, [377](#)
- CAP\_Capture\_PolarityEvent0, [378](#)
- CAP\_Capture\_PolarityEvent1, [378](#)
- CAP\_Capture\_PolarityEvent2, [378](#)
- CAP\_Capture\_PolarityEvent3, [378](#)
- CAP\_Capture\_Prescale, [379](#)
- CAP\_Capture\_RstEvent0, [379](#)
- CAP\_Capture\_RstEvent1, [379](#)
- CAP\_Capture\_RstEvent2, [379](#)
- CAP\_Capture\_RstEvent3, [379](#)
- CAP\_Capture\_StopVal, [379](#)
- CAP\_CaptureMode, [379](#)
- CAP\_Capture\_Mode\_Cycle
  - Типы, [73](#)
- niietcm4\_cap.h, [486](#)
- CAP\_Capture\_Mode\_Single
  - Типы, [73](#)
- niietcm4\_cap.h, [486](#)
- CAP\_Capture\_Mode\_TypeDef
  - Типы, [73](#)
- niietcm4\_cap.h, [486](#)
- CAP\_Capture\_Polarity\_NegEdge
  - Типы, [73](#)
- niietcm4\_cap.h, [486](#)
- CAP\_Capture\_Polarity\_PosEdge
  - Типы, [73](#)
- niietcm4\_cap.h, [486](#)
- CAP\_Capture\_Polarity\_TypeDef
  - Типы, [73](#)
- niietcm4\_cap.h, [486](#)
- CAP\_Capture\_PolarityEvent0
  - CAP\_Capture\_Init\_TypeDef, [378](#)
- CAP\_Capture\_PolarityEvent1
  - CAP\_Capture\_Init\_TypeDef, [378](#)
- CAP\_Capture\_PolarityEvent2
  - CAP\_Capture\_Init\_TypeDef, [378](#)
- CAP\_Capture\_PolarityEvent3
  - CAP\_Capture\_Init\_TypeDef, [378](#)
- CAP\_Capture\_Prescale
  - CAP\_Capture\_Init\_TypeDef, [379](#)
- CAP\_Capture\_RstEvent0
  - CAP\_Capture\_Init\_TypeDef, [379](#)
- CAP\_Capture\_RstEvent1
  - CAP\_Capture\_Init\_TypeDef, [379](#)
- CAP\_Capture\_RstEvent2
  - CAP\_Capture\_Init\_TypeDef, [379](#)
- CAP\_Capture\_RstEvent3
  - CAP\_Capture\_Init\_TypeDef, [379](#)
- CAP\_Capture\_SetCap0
  - Приватные функции, [269](#)

- Режим захвата, 92
- niietcm4\_cap.c, 471
- niietcm4\_cap.h, 489
- CAP\_Capture\_SetCap1
  - Приватные функции, 269
  - Режим захвата, 92
  - niietcm4\_cap.c, 471
  - niietcm4\_cap.h, 489
- CAP\_Capture\_SetCap2
  - Приватные функции, 269
  - Режим захвата, 92
  - niietcm4\_cap.c, 471
  - niietcm4\_cap.h, 490
- CAP\_Capture\_SetCap3
  - Приватные функции, 270
  - Режим захвата, 93
  - niietcm4\_cap.c, 471
  - niietcm4\_cap.h, 490
- CAP\_Capture\_StopVal
  - CAP\_Capture\_Init\_TypeDef, 379
- CAP\_Capture\_StructInit
  - Приватные функции, 270
  - Режим захвата, 93
  - niietcm4\_cap.c, 472
  - niietcm4\_cap.h, 490
- CAP\_CaptureMode
  - CAP\_Capture\_Init\_TypeDef, 379
- CAP\_DeInit
  - Конфигурация, 79
  - Приватные функции, 270
  - niietcm4\_cap.c, 472
  - niietcm4\_cap.h, 491
- CAP\_GetShadowTimer
  - Конфигурация, 79
  - Приватные функции, 271
  - niietcm4\_cap.c, 472
  - niietcm4\_cap.h, 491
- CAP\_GetTimer
  - Конфигурация, 80
  - Приватные функции, 271
  - niietcm4\_cap.c, 473
  - niietcm4\_cap.h, 491
- CAP\_Halt
  - CAP\_Init\_TypeDef, 380
- CAP\_Halt\_Free
  - Типы, 74
  - niietcm4\_cap.h, 487
- CAP\_Halt\_Stop
  - Типы, 73
  - niietcm4\_cap.h, 487
- CAP\_Halt\_StopOnZero
  - Типы, 73
  - niietcm4\_cap.h, 487
- CAP\_Halt\_TypeDef
  - Типы, 73
  - niietcm4\_cap.h, 487
- CAP\_ITCmd
  - Прерывания, 94
- Приватные функции, 271
- niietcm4\_cap.c, 473
- niietcm4\_cap.h, 492
- CAP\_ITForceCmd
  - Прерывания, 94
  - Приватные функции, 272
  - niietcm4\_cap.c, 474
  - niietcm4\_cap.h, 492
- CAP\_ITPendClear
  - Прерывания, 95
  - Приватные функции, 272
  - niietcm4\_cap.c, 474
  - niietcm4\_cap.h, 492
- CAP\_ITPendStatus
  - Прерывания, 95
  - Приватные функции, 272
  - niietcm4\_cap.c, 474
  - niietcm4\_cap.h, 493
- CAP\_ITSource\_All
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_CapEvent0
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_CapEvent1
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_CapEvent2
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_CapEvent3
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_GeneralInt
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_TimerEqCompare
  - Маски источников прерываний, 76
  - niietcm4\_cap.h, 484
- CAP\_ITSource\_TimerEqPeriod
  - Маски источников прерываний, 77
  - niietcm4\_cap.h, 485
- CAP\_ITSource\_TimerOvf
  - Маски источников прерываний, 77
  - niietcm4\_cap.h, 485
- CAP\_ITStatus
  - Прерывания, 95
  - Приватные функции, 273
  - niietcm4\_cap.c, 474
  - niietcm4\_cap.h, 493
- CAP\_ITStatusClear
  - Прерывания, 95
  - Приватные функции, 273
  - niietcm4\_cap.c, 475
  - niietcm4\_cap.h, 493
- CAP\_Init
  - Конфигурация, 80
  - Приватные функции, 271



- niietcm4\_cap.c, [473](#)
  - niietcm4\_cap.h, [491](#)
- CAP\_Init\_TypeDef, [380](#)
  - CAP\_Halt, [380](#)
  - CAP\_Mode, [380](#)
  - CAP\_SyncCmd, [380](#)
  - CAP\_SyncOut, [380](#)
- CAP\_Mode
  - CAP\_Init\_TypeDef, [380](#)
- 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, [487](#)
- CAP\_PWM\_Compare
  - CAP\_PWM\_Init\_TypeDef, [381](#)
- CAP\_PWM\_GetCompare
  - Приватные функции, [273](#)
  - Режим ШИМ, [84](#)
  - niietcm4\_cap.c, [475](#)
  - niietcm4\_cap.h, [493](#)
- CAP\_PWM\_GetPeriod
  - Приватные функции, [273](#)
  - Режим ШИМ, [84](#)
  - niietcm4\_cap.c, [475](#)
  - niietcm4\_cap.h, [494](#)
- CAP\_PWM\_GetShadowCompare
  - Приватные функции, [274](#)
  - Режим ШИМ, [85](#)
  - niietcm4\_cap.c, [475](#)
  - niietcm4\_cap.h, [494](#)
- CAP\_PWM\_GetShadowPeriod
  - Приватные функции, [274](#)
  - Режим ШИМ, [85](#)
  - niietcm4\_cap.c, [476](#)
  - niietcm4\_cap.h, [494](#)
- CAP\_PWM\_Init
  - Приватные функции, [274](#)
  - Режим ШИМ, [85](#)
  - niietcm4\_cap.c, [476](#)
  - niietcm4\_cap.h, [494](#)
- CAP\_PWM\_Init\_TypeDef, [381](#)
  - CAP\_PWM\_Compare, [381](#)
  - CAP\_PWM\_Period, [381](#)
  - CAP\_PWM\_Polarity, [381](#)
- CAP\_PWM\_Period
  - CAP\_PWM\_Init\_TypeDef, [381](#)
- CAP\_PWM\_Polarity
  - CAP\_PWM\_Init\_TypeDef, [381](#)
- 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, [275](#)
  - Режим ШИМ, [85](#)
  - niietcm4\_cap.c, [476](#)
  - niietcm4\_cap.h, [496](#)
- CAP\_PWM\_SetPeriod
  - Приватные функции, [275](#)
  - Режим ШИМ, [87](#)
  - niietcm4\_cap.c, [477](#)
  - niietcm4\_cap.h, [496](#)
- CAP\_PWM\_SetShadowCompare
  - Приватные функции, [275](#)
  - Режим ШИМ, [87](#)
  - niietcm4\_cap.c, [477](#)
  - niietcm4\_cap.h, [496](#)
- CAP\_PWM\_SetShadowPeriod
  - Приватные функции, [275](#)
  - Режим ШИМ, [87](#)
  - niietcm4\_cap.c, [477](#)
  - niietcm4\_cap.h, [497](#)
- CAP\_PWM\_StructInit
  - Приватные функции, [276](#)
  - Режим ШИМ, [88](#)
  - niietcm4\_cap.c, [477](#)
  - niietcm4\_cap.h, [497](#)
- CAP\_SetShadowTimer
  - Конфигурация, [80](#)
  - Приватные функции, [276](#)
  - niietcm4\_cap.c, [479](#)
  - niietcm4\_cap.h, [497](#)
- CAP\_SetTimer
  - Конфигурация, [80](#)
  - Приватные функции, [276](#)
  - niietcm4\_cap.c, [479](#)
  - niietcm4\_cap.h, [497](#)
- CAP\_StructInit
  - Конфигурация, [82](#)
  - Приватные функции, [277](#)
  - niietcm4\_cap.c, [479](#)
  - niietcm4\_cap.h, [498](#)
- CAP\_SwSync
  - Конфигурация, [82](#)
  - Приватные функции, [277](#)
  - niietcm4\_cap.c, [480](#)
  - niietcm4\_cap.h, [498](#)
- CAP\_SyncCmd
  - Конфигурация, [82](#)
  - Приватные функции, [277](#)
  - CAP\_Init\_TypeDef, [380](#)
  - niietcm4\_cap.c, [480](#)
  - niietcm4\_cap.h, [498](#)
- CAP\_SyncOut
  - CAP\_Init\_TypeDef, [380](#)
- 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, [480](#)
  - niietcm4\_cap.h, [498](#)
- CEMask
  - EXTMEM\_Init\_TypeDef, [389](#)
- CH
  - DMA\_ConfigStruct\_TypeDef, [386](#)
- CHANNEL\_CFG
  - DMA\_Channel\_TypeDef, [382](#)
- CHANNEL\_CFG\_CYCLE\_CTRL\_Msk
  - Маски для CHANNEL\_CFG, [98](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_CYCLE\_CTRL\_Pos
  - Маски для CHANNEL\_CFG, [98](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_DST\_INC\_Msk
  - Маски для CHANNEL\_CFG, [98](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_DST\_INC\_Pos
  - Маски для CHANNEL\_CFG, [98](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_DST\_PROT\_CTRL\_Msk
  - Маски для CHANNEL\_CFG, [98](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_DST\_PROT\_CTRL\_Pos
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_DST\_SIZE\_Msk
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_DST\_SIZE\_Pos
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_N\_MINUS\_1\_Msk
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [513](#)
- CHANNEL\_CFG\_N\_MINUS\_1\_Pos
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_NEXT\_USEBURST\_Msk
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_NEXT\_USEBURST\_Pos
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_R\_POWER\_Msk
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_R\_POWER\_Pos
  - Маски для CHANNEL\_CFG, [99](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_SRC\_INC\_Msk
  - Маски для CHANNEL\_CFG, [100](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_SRC\_INC\_Pos
  - Маски для CHANNEL\_CFG, [100](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_SRC\_PROT\_CTRL\_Msk
  - Маски для CHANNEL\_CFG, [100](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_SRC\_PROT\_CTRL\_Pos
  - Маски для CHANNEL\_CFG, [100](#)
  - niietcm4\_dma.h, [514](#)
- CHANNEL\_CFG\_SRC\_SIZE\_Msk
  - Маски для CHANNEL\_CFG, [100](#)
  - niietcm4\_dma.h, [515](#)
- CHANNEL\_CFG\_SRC\_SIZE\_Pos
  - Маски для CHANNEL\_CFG, [100](#)
  - niietcm4\_dma.h, [515](#)
- CHANNEL\_CFG\_bit
  - DMA\_Channel\_TypeDef, [382](#)
- CYCLE\_CTRL
  - \_CHANNEL\_CFG\_bits, [371](#)
- DMA, [279](#)
- DMA\_ArbitrationRate
  - DMA\_ChannelInit\_TypeDef, [383](#)
- DMA\_ArbitrationRate\_1
  - Типы, [112](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_1024
  - Типы, [113](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_128
  - Типы, [113](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_16
  - Типы, [112](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_2
  - Типы, [112](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_256
  - Типы, [113](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_32
  - Типы, [113](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_4
  - Типы, [112](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_512
  - Типы, [113](#)
  - niietcm4\_dma.h, [522](#)



- DMA\_ArbitrationRate\_64
  - Типы, [113](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_8
  - Типы, [112](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_ArbitrationRate\_TypeDef
  - Типы, [112](#)
  - niietcm4\_dma.h, [522](#)
- DMA\_BasePtrConfig
  - Конфигурация контроллера DMA, [120](#)
  - Приватные функции, [283](#)
  - niietcm4\_dma.c, [501](#)
  - niietcm4\_dma.h, [524](#)
- DMA\_Channel
  - DMA\_Init\_TypeDef, [387](#)
- DMA\_Channel\_0
  - Маски каналов по номеру, [102](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_1
  - Маски каналов по номеру, [102](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_10
  - Маски каналов по номеру, [102](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_11
  - Маски каналов по номеру, [102](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_12
  - Маски каналов по номеру, [102](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_13
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_14
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [515](#)
- DMA\_Channel\_15
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_16
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_17
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_18
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_19
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_2
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_20
  - Маски каналов по номеру, [103](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_21
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_22
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_23
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [516](#)
- DMA\_Channel\_3
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_4
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_5
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_6
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_7
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_8
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_9
  - Маски каналов по номеру, [104](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_ADCSEQ0
  - Маски каналов по имени, [106](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_ADCSEQ1
  - Маски каналов по имени, [106](#)
  - niietcm4\_dma.h, [517](#)
- DMA\_Channel\_ADCSEQ2
  - Маски каналов по имени, [106](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_ADCSEQ3
  - Маски каналов по имени, [106](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_ADCSEQ4
  - Маски каналов по имени, [106](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_ADCSEQ5
  - Маски каналов по имени, [107](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_ADCSEQ6
  - Маски каналов по имени, [107](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_ADCSEQ7
  - Маски каналов по имени, [107](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_All
  - Маски каналов DMA, [101](#)
  - niietcm4\_dma.h, [518](#)
- DMA\_Channel\_SPI0\_RX

- Маски каналов по имени, [107](#)
- [niietcm4\\_dma.h](#), [518](#)
- DMA\_Channel\_SPI0\_TX
  - Маски каналов по имени, [107](#)
  - [niietcm4\\_dma.h](#), [518](#)
- DMA\_Channel\_SPI1\_RX
  - Маски каналов по имени, [107](#)
  - [niietcm4\\_dma.h](#), [518](#)
- DMA\_Channel\_SPI1\_TX
  - Маски каналов по имени, [107](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_SPI2\_RX
  - Маски каналов по имени, [107](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_SPI2\_TX
  - Маски каналов по имени, [107](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_SPI3\_RX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_SPI3\_TX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_TypeDef, [382](#)
  - CHANNEL\_CFG, [382](#)
  - CHANNEL\_CFG\_bit, [382](#)
  - DST\_DATA\_END, [382](#)
  - SRC\_DATA\_END, [382](#)
- DMA\_Channel\_UART0\_RX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_UART0\_TX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_UART1\_RX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_UART1\_TX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [519](#)
- DMA\_Channel\_UART2\_RX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [520](#)
- DMA\_Channel\_UART2\_TX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [520](#)
- DMA\_Channel\_UART3\_RX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [520](#)
- DMA\_Channel\_UART3\_TX
  - Маски каналов по имени, [108](#)
  - [niietcm4\\_dma.h](#), [520](#)
- DMA\_ChannelDeInit
  - Инициализация каналов DMA, [116](#)
  - Приватные функции, [284](#)
  - [niietcm4\\_dma.c](#), [502](#)
  - [niietcm4\\_dma.h](#), [525](#)
- DMA\_ChannelEnable
  - DMA\_Init\_TypeDef, [387](#)
- DMA\_ChannelEnableCmd
  - Конфигурация контроллера DMA, [120](#)
  - Приватные функции, [284](#)
  - [niietcm4\\_dma.c](#), [502](#)
  - [niietcm4\\_dma.h](#), [525](#)
- DMA\_ChannelInit
  - Инициализация каналов DMA, [116](#)
  - Приватные функции, [284](#)
  - [niietcm4\\_dma.c](#), [502](#)
  - [niietcm4\\_dma.h](#), [525](#)
- DMA\_ChannelInit\_TypeDef, [383](#)
  - DMA\_ArbitrationRate, [383](#)
  - DMA\_DstDataEndPtr, [383](#)
  - DMA\_DstDataInc, [383](#)
  - DMA\_DstDataSize, [384](#)
  - DMA\_DstProtect, [384](#)
  - DMA\_Mode, [384](#)
  - DMA\_NextUseburst, [384](#)
  - DMA\_SrcDataEndPtr, [384](#)
  - DMA\_SrcDataInc, [384](#)
  - DMA\_SrcDataSize, [384](#)
  - DMA\_SrcProtect, [385](#)
  - DMA\_TransfersTotal, [385](#)
- DMA\_ChannelStructInit
  - Инициализация каналов DMA, [117](#)
  - Приватные функции, [285](#)
  - [niietcm4\\_dma.c](#), [503](#)
  - [niietcm4\\_dma.h](#), [526](#)
- DMA\_ClearErrorStatus
  - Приватные функции, [285](#)
  - Статусная информация, [124](#)
  - [niietcm4\\_dma.c](#), [503](#)
  - [niietcm4\\_dma.h](#), [526](#)
- DMA\_ConfigData\_TypeDef, [385](#)
  - ALT\_DATA, [386](#)
  - PRM\_DATA, [386](#)
  - RESERVED0, [386](#)
  - RESERVED1, [386](#)
- DMA\_ConfigStruct\_TypeDef, [386](#)
  - CH, [386](#)
- DMA\_DataInc\_16
  - Типы, [113](#)
  - [niietcm4\\_dma.h](#), [522](#)
- DMA\_DataInc\_32
  - Типы, [113](#)
  - [niietcm4\\_dma.h](#), [522](#)
- DMA\_DataInc\_8
  - Типы, [113](#)
  - [niietcm4\\_dma.h](#), [522](#)
- DMA\_DataInc\_Disable
  - Типы, [113](#)
  - [niietcm4\\_dma.h](#), [522](#)
- DMA\_DataInc\_TypeDef
  - Типы, [113](#)
  - [niietcm4\\_dma.h](#), [522](#)
- DMA\_DataSize\_16
  - Типы, [113](#)

- niietcm4\_dma.h, 523
- DMA\_DataSize\_32
  - Типы, 113
- niietcm4\_dma.h, 523
- DMA\_DataSize\_8
  - Типы, 113
- niietcm4\_dma.h, 523
- DMA\_DataSize\_TypeDef
  - Типы, 113
- niietcm4\_dma.h, 522
- DMA\_DeInit
  - Инициализация контроллера DMA, 118
  - Приватные функции, 285
  - niietcm4\_dma.c, 503
  - niietcm4\_dma.h, 526
- DMA\_DstDataEndPtr
  - DMA\_ChannelInit\_TypeDef, 383
- DMA\_DstDataInc
  - DMA\_ChannelInit\_TypeDef, 383
- DMA\_DstDataSize
  - DMA\_ChannelInit\_TypeDef, 384
- DMA\_DstProtect
  - DMA\_ChannelInit\_TypeDef, 384
- DMA\_ErrorStatus
  - Приватные функции, 287
  - Статусная информация, 124
  - niietcm4\_dma.c, 505
  - niietcm4\_dma.h, 528
- DMA\_HighPriority
  - DMA\_Init\_TypeDef, 387
- DMA\_HighPriorityCmd
  - Конфигурация контроллера DMA, 121
  - Приватные функции, 287
  - niietcm4\_dma.c, 505
  - niietcm4\_dma.h, 528
- DMA\_Init
  - Инициализация контроллера DMA, 118
  - Приватные функции, 287
  - niietcm4\_dma.c, 505
  - niietcm4\_dma.h, 528
- DMA\_Init\_TypeDef, 387
  - DMA\_Channel, 387
  - DMA\_ChannelEnable, 387
  - DMA\_HighPriority, 387
  - DMA\_PrmAlt, 387
  - DMA\_Protection, 387
  - DMA\_ReqMask, 388
  - DMA\_UseBurst, 388
- DMA\_MasterEnableCmd
  - Конфигурация контроллера DMA, 121
  - Приватные функции, 288
  - niietcm4\_dma.c, 506
  - niietcm4\_dma.h, 529
- DMA\_MasterEnableStatus
  - Приватные функции, 288
  - Статусная информация, 124
  - niietcm4\_dma.c, 506
  - niietcm4\_dma.h, 529
- DMA\_Mode
  - DMA\_ChannelInit\_TypeDef, 384
- DMA\_Mode\_AltMemScatGath
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_AltPeriphScatGath
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_AutoReq
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_Basic
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_Disable
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_PingPong
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_PrmMemScatGath
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_PrmPeriphScatGath
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_Mode\_TypeDef
  - Типы, 113
  - niietcm4\_dma.h, 523
- DMA\_NextUseburst
  - DMA\_ChannelInit\_TypeDef, 384
- DMA\_PrmAlt
  - DMA\_Init\_TypeDef, 387
- DMA\_PrmAltCmd
  - Конфигурация контроллера DMA, 121
  - Приватные функции, 288
  - niietcm4\_dma.c, 506
  - niietcm4\_dma.h, 529
- DMA\_Protect\_TypeDef, 388
  - BUFFERABLE, 388
  - CACHEABLE, 388
  - PRIVELGED, 389
- DMA\_Protection
  - DMA\_Init\_TypeDef, 387
- DMA\_ProtectionConfig
  - Конфигурация контроллера DMA, 122
  - Приватные функции, 288
  - niietcm4\_dma.c, 506
  - niietcm4\_dma.h, 529
- DMA\_ReqMask
  - DMA\_Init\_TypeDef, 388
- DMA\_ReqMaskCmd
  - Конфигурация контроллера DMA, 122
  - Приватные функции, 289
  - niietcm4\_dma.c, 507
  - niietcm4\_dma.h, 530
- DMA\_SWRequestCmd
  - Конфигурация контроллера DMA, 122

- Приватные функции, [290](#)
- niietcm4\_dma.c, [508](#)
- niietcm4\_dma.h, [531](#)
- DMA\_SrcDataEndPtr
  - DMA\_ChannelInit\_TypeDef, [384](#)
- DMA\_SrcDataInc
  - DMA\_ChannelInit\_TypeDef, [384](#)
- DMA\_SrcDataSize
  - DMA\_ChannelInit\_TypeDef, [384](#)
- DMA\_SrcProtect
  - DMA\_ChannelInit\_TypeDef, [385](#)
- DMA\_State\_Done
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_Free
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_Pause
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_PeriphScatGath
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_ReadConfigData
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_ReadDstDataEndPtr
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_ReadSrcData
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_ReadSrcDataEndPtr
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_TypeDef
  - Типы, [113](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_WaitReq
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_WriteConfigData
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_State\_WriteDstData
  - Типы, [114](#)
  - niietcm4\_dma.h, [523](#)
- DMA\_StateStatus
  - Приватные функции, [289](#)
  - Статусная информация, [124](#)
  - niietcm4\_dma.c, [507](#)
  - niietcm4\_dma.h, [530](#)
- DMA\_StructInit
  - Инициализация контроллера DMA, [118](#)
  - Приватные функции, [289](#)
  - niietcm4\_dma.c, [507](#)
  - niietcm4\_dma.h, [530](#)
- DMA\_TransfersTotal
  - DMA\_ChannelInit\_TypeDef, [385](#)
- DMA\_UseBurst
  - DMA\_Init\_TypeDef, [388](#)
- DMA\_UseBurstCmd
  - Конфигурация контроллера DMA, [123](#)
  - Приватные функции, [290](#)
  - niietcm4\_dma.c, [508](#)
  - niietcm4\_dma.h, [531](#)
- DMA\_WaitOnReqStatus
  - Приватные функции, [290](#)
  - Статусная информация, [125](#)
  - niietcm4\_dma.c, [508](#)
  - niietcm4\_dma.h, [531](#)
- DST\_DATA\_END
  - DMA\_Channel\_TypeDef, [382](#)
- DST\_INC
  - \_CHANNEL\_CFG\_bits, [371](#)
- DST\_PROT\_BUFFERABLE
  - \_CHANNEL\_CFG\_bits, [372](#)
- DST\_PROT\_CACHEABLE
  - \_CHANNEL\_CFG\_bits, [372](#)
- DST\_PROT\_PRIVILEGED
  - \_CHANNEL\_CFG\_bits, [372](#)
- DST\_SIZE
  - \_CHANNEL\_CFG\_bits, [372](#)
- EXT\_MEM\_CFG\_Reset\_Value
  - Начальные значения регистров, [294](#)
  - niietcm4\_extmem.c, [533](#)
- EXT\_OSC\_VALUE
  - Настройка драйвера, [364](#)
  - niietcm4.h, [401](#)
- EXTMEM, [291](#)
- EXTMEM\_CEMask\_Addr\_11
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [535](#)
- EXTMEM\_CEMask\_Addr\_11\_19
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [535](#)
- EXTMEM\_CEMask\_Addr\_12
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [536](#)
- EXTMEM\_CEMask\_Addr\_13
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [536](#)
- EXTMEM\_CEMask\_Addr\_14
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [536](#)
- EXTMEM\_CEMask\_Addr\_15
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [536](#)
- EXTMEM\_CEMask\_Addr\_16
  - Маски адреса, [127](#)
  - niietcm4\_extmem.h, [536](#)
- EXTMEM\_CEMask\_Addr\_17
  - Маски адреса, [128](#)
  - niietcm4\_extmem.h, [536](#)
- EXTMEM\_CEMask\_Addr\_18
  - Маски адреса, [128](#)

- niietcm4\_extmem.h, 536
- EXTMEM\_CEMask\_Addr\_19
  - Маски адреса, 128
  - niietcm4\_extmem.h, 536
- EXTMEM\_DeInit
  - Функции, 133
  - Приватные функции, 295
  - niietcm4\_extmem.c, 533
  - niietcm4\_extmem.h, 539
- EXTMEM\_Init
  - Функции, 133
  - Приватные функции, 295
  - niietcm4\_extmem.c, 533
  - niietcm4\_extmem.h, 539
- EXTMEM\_Init\_TypeDef, 389
  - CEMask, 389
  - EXTMEM\_RWWaitState, 390
  - EXTMEM\_ReadWaitState, 389
  - EXTMEM\_Width, 390
  - EXTMEM\_WriteWaitState, 390
- EXTMEM\_RWWaitState
  - EXTMEM\_Init\_TypeDef, 390
- EXTMEM\_RWWaitState\_1
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_2
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_3
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_4
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_5
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_6
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_7
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_8
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_RWWaitState\_TypeDef
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState
  - EXTMEM\_Init\_TypeDef, 389
- EXTMEM\_ReadWaitState\_1
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_2
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_3
  - Типы, 131
  - niietcm4\_extmem.h, 538
- Типы, 131
- niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_4
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_5
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_6
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_7
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_8
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_ReadWaitState\_TypeDef
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_StructInit
  - Функции, 133
  - Приватные функции, 295
  - niietcm4\_extmem.c, 533
  - niietcm4\_extmem.h, 539
- EXTMEM\_Width
  - EXTMEM\_Init\_TypeDef, 390
- EXTMEM\_Width\_16bit
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_Width\_8bit
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_Width\_TypeDef
  - Типы, 131
  - niietcm4\_extmem.h, 538
- EXTMEM\_WriteWaitState
  - EXTMEM\_Init\_TypeDef, 390
- EXTMEM\_WriteWaitState\_1
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_2
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_3
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_4
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_5
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_6
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_7
  - Типы, 132

- niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_8
  - Типы, 132
  - niietcm4\_extmem.h, 539
- EXTMEM\_WriteWaitState\_TypeDef
  - Типы, 131
  - niietcm4\_extmem.h, 538
- GPIO, 297
- GPIO\_AltFunc
  - GPIO\_Init\_TypeDef, 391
- GPIO\_AltFunc\_1
  - Типы, 139
  - niietcm4\_gpio.h, 560
- GPIO\_AltFunc\_2
  - Типы, 139
  - niietcm4\_gpio.h, 560
- GPIO\_AltFunc\_3
  - Типы, 139
  - niietcm4\_gpio.h, 560
- GPIO\_AltFunc\_TypeDef
  - Типы, 138
  - niietcm4\_gpio.h, 560
- GPIO\_AltFuncConfig
  - Инициализация и деинициализация, 148
  - Приватные функции, 306
  - niietcm4\_gpio.c, 545
  - niietcm4\_gpio.h, 563
- GPIO\_ClearBits
  - Битовые операции, 155
  - Приватные функции, 307
  - niietcm4\_gpio.c, 545
  - niietcm4\_gpio.h, 564
- GPIO\_DATAOUT\_Reset\_Value
  - Начальные значения регистров, 300
  - niietcm4\_gpio.c, 542
- GPIO\_DeInit
  - Инициализация и деинициализация, 148
  - Приватные функции, 307
  - niietcm4\_gpio.c, 545
  - niietcm4\_gpio.h, 564
- GPIO\_Dir
  - GPIO\_Init\_TypeDef, 391
- GPIO\_Dir\_In
  - Типы, 139
  - niietcm4\_gpio.h, 560
- GPIO\_Dir\_Out
  - Типы, 139
  - niietcm4\_gpio.h, 560
- GPIO\_Dir\_TypeDef
  - Типы, 139
  - niietcm4\_gpio.h, 560
- GPIO\_GPIODEN0\_Reset\_Value
  - Начальные значения регистров, 300
  - niietcm4\_gpio.c, 542
- 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, 543
- GPIO\_GPIOPCTLx\_Reset\_Value
  - Начальные значения регистров, 301
  - niietcm4\_gpio.c, 543
- GPIO\_GPIOPUCTLx\_Reset\_Value
  - Начальные значения регистров, 301
  - niietcm4\_gpio.c, 543
- GPIO\_GPIOQEx\_Reset\_Value
  - Начальные значения регистров, 301
  - niietcm4\_gpio.c, 543
- 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, 159
  - Приватные функции, 308
  - niietcm4\_gpio.c, 546
  - niietcm4\_gpio.h, 565
- GPIO\_ITConfig
  - Прерывания, 159
  - Приватные функции, 308
  - niietcm4\_gpio.c, 546
  - niietcm4\_gpio.h, 565
- GPIO\_ITStatusClear
  - Прерывания, 159
  - Приватные функции, 309
  - niietcm4\_gpio.c, 547
  - niietcm4\_gpio.h, 566
- GPIO\_Init
  - Инициализация и деинициализация, 148
  - Приватные функции, 307
  - niietcm4\_gpio.c, 546
  - niietcm4\_gpio.h, 564
- GPIO\_Init\_TypeDef, 390
  - GPIO\_AltFunc, 391
  - GPIO\_Dir, 391
  - GPIO\_Load, 391
  - GPIO\_Mode, 391
  - GPIO\_Out, 391
  - GPIO\_OutMode, 391
  - GPIO\_Pin, 391
  - GPIO\_PullUp, 392



- GPIO\_IntPol\_Neg
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_IntPol\_Pos
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_IntPol\_TypeDef
  - Типы, [139](#)
  - niietcm4\_gpio.h, [560](#)
- GPIO\_IntType\_Edge
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_IntType\_Level
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_IntType\_TypeDef
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Load
  - GPIO\_Init\_TypeDef, [391](#)
- GPIO\_Load\_16mA
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Load\_8mA
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Load\_TypeDef
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Mode
  - GPIO\_Init\_TypeDef, [391](#)
- GPIO\_Mode\_AltFunc
  - Типы, [140](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Mode\_IO
  - Типы, [140](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Mode\_TypeDef
  - Типы, [139](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Out
  - GPIO\_Init\_TypeDef, [391](#)
- GPIO\_Out\_Dis
  - Типы, [140](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Out\_En
  - Типы, [140](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Out\_TypeDef
  - Типы, [140](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_OutMode
  - GPIO\_Init\_TypeDef, [391](#)
- GPIO\_OutMode\_OD
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_OutMode\_PP
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- niietcm4\_gpio.h, [562](#)
- GPIO\_OutMode\_TypeDef
  - Типы, [140](#)
  - niietcm4\_gpio.h, [561](#)
- GPIO\_Pin
  - GPIO\_Init\_TypeDef, [391](#)
- GPIO\_Pin\_0
  - Маски пинов, [143](#)
  - niietcm4\_gpio.h, [554](#)
- GPIO\_Pin\_0\_3
  - Маски пинов, [143](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_0\_7
  - Маски пинов, [143](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_1
  - Маски пинов, [143](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_10
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_11
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_12
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_12\_15
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_13
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_14
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [555](#)
- GPIO\_Pin\_15
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_2
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_3
  - Маски пинов, [144](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_4
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_4\_7
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_5
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_6
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_7

- Маски пинов, [145](#)
- niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_8
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [556](#)
- GPIO\_Pin\_8\_11
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [557](#)
- GPIO\_Pin\_8\_15
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [557](#)
- GPIO\_Pin\_9
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [557](#)
- GPIO\_Pin\_All
  - Маски пинов, [145](#)
  - niietcm4\_gpio.h, [557](#)
- GPIO\_PullUp
  - GPIO\_Init\_TypeDef, [392](#)
- GPIO\_PullUp\_Dis
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_PullUp\_En
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_PullUp\_TypeDef
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_Qual\_Dis
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_Qual\_En
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_Qual\_TypeDef
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_QualCmd
  - Фильтрация, [157](#)
  - Приватные функции, [309](#)
  - niietcm4\_gpio.c, [547](#)
  - niietcm4\_gpio.h, [566](#)
- GPIO\_QualConfig
  - Фильтрация, [157](#)
  - Приватные функции, [309](#)
  - niietcm4\_gpio.c, [547](#)
  - niietcm4\_gpio.h, [566](#)
- GPIO\_QualMode\_3sample
  - Типы, [141](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_QualMode\_6sample
  - Типы, [141](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_QualMode\_TypeDef
  - Типы, [140](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_Read
  - Чтение, [151](#)
- Приватные функции, [310](#)
- niietcm4\_gpio.c, [548](#)
- niietcm4\_gpio.h, [567](#)
- GPIO\_ReadBit
  - Чтение, [151](#)
  - Приватные функции, [310](#)
  - niietcm4\_gpio.c, [548](#)
  - niietcm4\_gpio.h, [567](#)
- GPIO\_ReadMask
  - Чтение, [151](#)
  - Приватные функции, [310](#)
  - niietcm4\_gpio.c, [548](#)
  - niietcm4\_gpio.h, [567](#)
- 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, [544](#)
- GPIO\_Regs\_GPIOA\_Mask
  - Маски портов, [303](#)
  - niietcm4\_gpio.c, [544](#)
- GPIO\_Regs\_GPIOB\_Mask
  - Маски портов, [303](#)
  - niietcm4\_gpio.c, [544](#)
- GPIO\_Regs\_GPIOC\_Mask
  - Маски портов, [303](#)
  - niietcm4\_gpio.c, [544](#)
- 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, [155](#)
  - Приватные функции, [311](#)
  - niietcm4\_gpio.c, [549](#)
  - niietcm4\_gpio.h, [568](#)
- GPIO\_StructInit
  - Инициализация и деинициализация, [149](#)
  - Приватные функции, [311](#)
  - niietcm4\_gpio.c, [549](#)
  - niietcm4\_gpio.h, [568](#)
- GPIO\_Sync\_Dis
  - Типы, [141](#)
  - niietcm4\_gpio.h, [562](#)
- GPIO\_Sync\_En
  - Типы, [141](#)



- niietcm4\_gpio.h, 562
- GPIO\_Sync\_TypeDef
  - Типы, 141
- niietcm4\_gpio.h, 562
- GPIO\_SyncCmd
  - Фильтрация, 158
  - Приватные функции, 311
- niietcm4\_gpio.c, 549
  - niietcm4\_gpio.h, 568
- GPIO\_ToggleBits
  - Битовые операции, 155
  - Приватные функции, 312
- niietcm4\_gpio.c, 550
  - niietcm4\_gpio.h, 569
- GPIO\_Write
  - Приватные функции, 312
  - Запись, 153
- niietcm4\_gpio.c, 550
  - niietcm4\_gpio.h, 569
- GPIO\_WriteBit
  - Приватные функции, 312
  - Запись, 153
- niietcm4\_gpio.c, 550
  - niietcm4\_gpio.h, 569
- GPIO\_WriteMask
  - Приватные функции, 313
  - Запись, 153
- niietcm4\_gpio.c, 551
  - niietcm4\_gpio.h, 570
- INT\_OSC\_VALUE
  - Настройка драйвера, 364
- niietcm4.h, 401
- IS\_ADC\_AVERAGE
  - Типы, 28
- niietcm4\_adc.h, 433
- IS\_ADC\_DC\_CHANNEL
  - Типы, 28
- niietcm4\_adc.h, 433
- IS\_ADC\_DC\_CONDITION
  - Типы, 29
- niietcm4\_adc.h, 434
- IS\_ADC\_DC\_MODE
  - Типы, 29
- niietcm4\_adc.h, 434
- IS\_ADC\_DC\_MODULE
  - Типы, 29
- niietcm4\_adc.h, 434
- IS\_ADC\_MEASURE
  - Типы, 30
- niietcm4\_adc.h, 435
- IS\_ADC\_MODE
  - Типы, 30
- niietcm4\_adc.h, 435
- IS\_ADC\_MODULE
  - Типы, 30
- niietcm4\_adc.h, 435
- IS\_ADC\_RESOLUTION
  - Типы, 30
- niietcm4\_adc.h, 435
- IS\_ADC\_SEQ\_FIFO\_LEVEL
  - Типы, 31
- niietcm4\_adc.h, 436
- IS\_ADC\_SEQ\_MODULE
  - Типы, 31
- niietcm4\_adc.h, 436
- IS\_ADC\_SEQ\_START\_EVENT
  - Типы, 31
- niietcm4\_adc.h, 436
- IS\_BOOTFLASH\_STATUS
  - Типы, 65
- niietcm4\_bootflash.h, 464
- IS\_CAP\_ALL\_PERIPH
  - Типы, 366
- niietcm4.h, 401
- IS\_CAP\_CAPTURE\_MODE
  - Типы, 72
- niietcm4\_cap.h, 485
- IS\_CAP\_CAPTURE\_POLARITY
  - Типы, 72
- niietcm4\_cap.h, 485
- IS\_CAP\_HALT
  - Типы, 72
- niietcm4\_cap.h, 485
- IS\_CAP\_MODE
  - Типы, 72
- niietcm4\_cap.h, 485
- IS\_CAP\_PWM\_POLARITY
  - Типы, 72
- niietcm4\_cap.h, 486
- IS\_CAP\_SYNC\_OUT
  - Типы, 73
- niietcm4\_cap.h, 486
- IS\_DMA\_ARBITRATION\_RATE
  - Типы, 111
- niietcm4\_dma.h, 520
- IS\_DMA\_DATA\_INC
  - Типы, 111
- niietcm4\_dma.h, 520
- IS\_DMA\_DATA\_SIZE
  - Типы, 111
- niietcm4\_dma.h, 521
- IS\_DMA\_MODE
  - Типы, 112
- niietcm4\_dma.h, 521
- IS\_DMA\_STATE
  - Типы, 112
- niietcm4\_dma.h, 521
- IS\_EXTMEM\_READ\_WAITSTATE
  - Типы, 129
- niietcm4\_extmem.h, 536
- IS\_EXTMEM\_RW\_WAITSTATE
  - Типы, 130
- niietcm4\_extmem.h, 537
- IS\_EXTMEM\_WIDTH
  - Типы, 130
- niietcm4\_extmem.h, 537

- IS\_EXTMEM\_WRITE\_WAITSTATE
  - Типы, [130](#)
  - niietcm4\_extmem.h, [537](#)
- IS\_GET\_DMA\_CHANNEL
  - Маски каналов DMA, [101](#)
  - niietcm4\_dma.h, [521](#)
- IS\_GET\_GPIO\_PIN
  - Маски пинов, [146](#)
  - niietcm4\_gpio.h, [557](#)
- IS\_GPIO\_ALL\_PERIPH
  - Типы, [366](#)
  - niietcm4.h, [401](#)
- IS\_GPIO\_ALT\_FUNC
  - Типы, [136](#)
  - niietcm4\_gpio.h, [557](#)
- IS\_GPIO\_DIR
  - Типы, [136](#)
  - niietcm4\_gpio.h, [558](#)
- IS\_GPIO\_INT\_POL
  - Типы, [136](#)
  - niietcm4\_gpio.h, [558](#)
- IS\_GPIO\_INT\_TYPE
  - Типы, [136](#)
  - niietcm4\_gpio.h, [558](#)
- IS\_GPIO\_LOAD
  - Типы, [137](#)
  - niietcm4\_gpio.h, [558](#)
- IS\_GPIO\_MODE
  - Типы, [137](#)
  - niietcm4\_gpio.h, [558](#)
- IS\_GPIO\_OUT
  - Типы, [137](#)
  - niietcm4\_gpio.h, [559](#)
- IS\_GPIO\_OUT\_MODE
  - Типы, [137](#)
  - niietcm4\_gpio.h, [559](#)
- IS\_GPIO\_PULLUP
  - Типы, [137](#)
  - niietcm4\_gpio.h, [559](#)
- IS\_GPIO\_QUAL
  - Типы, [138](#)
  - niietcm4\_gpio.h, [559](#)
- IS\_GPIO\_QUAL\_MODE
  - Типы, [138](#)
  - niietcm4\_gpio.h, [559](#)
- IS\_GPIO\_SYNC
  - Типы, [138](#)
  - niietcm4\_gpio.h, [560](#)
- IS\_RCC\_ADC\_CLK
  - Типы, [164](#)
  - niietcm4\_rcc.h, [584](#)
- IS\_RCC\_PERIPH\_CLK
  - Типы, [164](#)
  - niietcm4\_rcc.h, [584](#)
- IS\_RCC\_PLL\_NO
  - Типы, [164](#)
  - niietcm4\_rcc.h, [585](#)
- IS\_RCC\_PLL\_REF
  - Типы, [164](#)
  - niietcm4\_rcc.h, [585](#)
- IS\_RCC\_SPI\_CLK
  - Типы, [165](#)
  - niietcm4\_rcc.h, [585](#)
- IS\_RCC\_SYS\_CLK
  - Типы, [165](#)
  - niietcm4\_rcc.h, [585](#)
- IS\_RCC\_UART\_CLK
  - Типы, [165](#)
  - niietcm4\_rcc.h, [585](#)
- IS\_RCC\_USB\_CLK
  - Типы, [165](#)
  - niietcm4\_rcc.h, [586](#)
- IS\_RCC\_USB\_FREQ
  - Типы, [166](#)
  - niietcm4\_rcc.h, [586](#)
- IS\_RTC\_FORMAT
  - Типы, [185](#)
  - niietcm4\_rtc.h, [600](#)
- IS\_RTC\_MONTH
  - Типы, [185](#)
  - niietcm4\_rtc.h, [600](#)
- IS\_RTC\_WEEKDAY
  - Типы, [185](#)
  - niietcm4\_rtc.h, [601](#)
- IS\_SPI\_ALL\_PERIPH
  - Типы, [367](#)
  - niietcm4.h, [402](#)
- IS\_TIMER\_ALL\_PERIPH
  - Типы, [367](#)
  - niietcm4.h, [402](#)
- IS\_TIMER\_EXT\_INPUT
  - Типы, [188](#)
  - niietcm4\_timer.h, [608](#)
- IS\_UART\_ALL\_PERIPH
  - Типы, [367](#)
  - niietcm4.h, [402](#)
- IS\_UART\_DATA\_WIDTH
  - Типы, [198](#)
  - niietcm4\_uart.h, [625](#)
- IS\_UART\_DIR
  - Типы, [198](#)
  - niietcm4\_uart.h, [625](#)
- IS\_UART\_FIFO\_LEVEL
  - Типы, [198](#)
  - niietcm4\_uart.h, [626](#)
- IS\_UART\_PARITY\_BIT
  - Типы, [198](#)
  - niietcm4\_uart.h, [626](#)
- IS\_UART\_STOP\_BIT
  - Типы, [198](#)
  - niietcm4\_uart.h, [626](#)
- IS\_USERFLASH\_STATUS
  - Типы, [223](#)
  - niietcm4\_userflash.h, [643](#)
- N\_MINUS\_1
  - \_CHANNEL\_CFG\_bits, [372](#)

- NEXT\_USEBURST  
  CHANNEL\_CFG\_bits, 372
- niietcm4.h, 399
- EXT\_OSC\_VALUE, 401
  - INT\_OSC\_VALUE, 401
  - IS\_CAP\_ALL\_PERIPH, 401
  - IS\_GPIO\_ALL\_PERIPH, 401
  - IS\_SPI\_ALL\_PERIPH, 402
  - IS\_TIMER\_ALL\_PERIPH, 402
  - IS\_UART\_ALL\_PERIPH, 402
- niietcm4\_adc.c, 403
- ADC\_Cmd, 405
  - ADC\_DC\_Cmd, 406
  - ADC\_DC\_DeInit, 406
  - ADC\_DC\_GetLastData, 406
  - ADC\_DC\_ITCmd, 407
  - ADC\_DC\_ITConfig, 407
  - ADC\_DC\_ITGenCmd, 408
  - ADC\_DC\_ITMaskCmd, 408
  - ADC\_DC\_ITMaskedStatus, 408
  - ADC\_DC\_ITRawStatus, 409
  - ADC\_DC\_ITStatusClear, 409
  - ADC\_DC\_Init, 407
  - ADC\_DC\_StructInit, 409
  - ADC\_DC\_TrigStatus, 410
  - ADC\_DC\_TrigStatusClear, 410
  - ADC\_DeInit, 410
  - ADC\_Init, 410
  - ADC\_SEQ\_Cmd, 412
  - ADC\_SEQ\_DMACmd, 412
  - ADC\_SEQ\_DMAConfig, 413
  - ADC\_SEQ\_DMAErrorStatus, 413
  - ADC\_SEQ\_DMAErrorStatusClear, 413
  - ADC\_SEQ\_DeInit, 412
  - ADC\_SEQ\_FIFOEmptyStatus, 414
  - ADC\_SEQ\_FIFOEmptyStatusClear, 414
  - ADC\_SEQ\_FIFOFullStatus, 414
  - ADC\_SEQ\_FIFOFullStatusClear, 415
  - ADC\_SEQ\_GetConversionCount, 415
  - ADC\_SEQ\_GetFIFOData, 415
  - ADC\_SEQ\_GetFIFOLoad, 415
  - ADC\_SEQ\_GetITCount, 417
  - ADC\_SEQ\_ITCmd, 417
  - ADC\_SEQ\_ITConfig, 418
  - ADC\_SEQ\_ITCountRst, 418
  - ADC\_SEQ\_ITMaskedStatus, 418
  - ADC\_SEQ\_ITRawStatus, 419
  - ADC\_SEQ\_ITStatusClear, 419
  - ADC\_SEQ\_Init, 417
  - ADC\_SEQ\_SWReq, 420
  - ADC\_SEQ\_StructInit, 419
  - ADC\_StructInit, 420
- niietcm4\_adc.h, 420
- ADC\_Average\_16, 437
  - ADC\_Average\_2, 437
  - ADC\_Average\_32, 437
  - ADC\_Average\_4, 437
  - ADC\_Average\_64, 437
  - ADC\_Average\_8, 437
  - ADC\_Average\_Disable, 437
  - ADC\_Average\_TypeDef, 437
  - ADC\_Channel\_0, 427
  - ADC\_Channel\_1, 427
  - ADC\_Channel\_10, 427
  - ADC\_Channel\_11, 427
  - ADC\_Channel\_12, 427
  - ADC\_Channel\_13, 427
  - ADC\_Channel\_14, 427
  - ADC\_Channel\_15, 427
  - ADC\_Channel\_16, 427
  - ADC\_Channel\_17, 428
  - ADC\_Channel\_18, 428
  - ADC\_Channel\_19, 428
  - ADC\_Channel\_2, 428
  - ADC\_Channel\_20, 428
  - ADC\_Channel\_21, 428
  - ADC\_Channel\_22, 428
  - ADC\_Channel\_23, 428
  - ADC\_Channel\_3, 428
  - ADC\_Channel\_4, 429
  - ADC\_Channel\_5, 429
  - ADC\_Channel\_6, 429
  - ADC\_Channel\_7, 429
  - ADC\_Channel\_8, 429
  - ADC\_Channel\_9, 429
  - ADC\_Channel\_All, 429
  - ADC\_Channel\_None, 429
  - ADC\_Cmd, 441
  - ADC\_DC\_0, 429
  - ADC\_DC\_1, 429
  - ADC\_DC\_10, 430
  - ADC\_DC\_11, 430
  - ADC\_DC\_12, 430
  - ADC\_DC\_13, 430
  - ADC\_DC\_14, 430
  - ADC\_DC\_15, 430
  - ADC\_DC\_16, 430
  - ADC\_DC\_17, 430
  - ADC\_DC\_18, 430
  - ADC\_DC\_19, 431
  - ADC\_DC\_2, 431
  - ADC\_DC\_20, 431
  - ADC\_DC\_21, 431
  - ADC\_DC\_22, 431
  - ADC\_DC\_23, 431
  - ADC\_DC\_3, 431
  - ADC\_DC\_4, 431
  - ADC\_DC\_5, 431
  - ADC\_DC\_6, 431
  - ADC\_DC\_7, 432
  - ADC\_DC\_8, 432
  - ADC\_DC\_9, 432
  - ADC\_DC\_All, 432
  - ADC\_DC\_Channel\_0, 437
  - ADC\_DC\_Channel\_1, 437
  - ADC\_DC\_Channel\_10, 438

- ADC\_DC\_Channel\_11, [438](#)
- ADC\_DC\_Channel\_12, [438](#)
- ADC\_DC\_Channel\_13, [438](#)
- ADC\_DC\_Channel\_14, [438](#)
- ADC\_DC\_Channel\_15, [438](#)
- ADC\_DC\_Channel\_16, [438](#)
- ADC\_DC\_Channel\_17, [438](#)
- ADC\_DC\_Channel\_18, [438](#)
- ADC\_DC\_Channel\_19, [438](#)
- ADC\_DC\_Channel\_2, [437](#)
- ADC\_DC\_Channel\_20, [438](#)
- ADC\_DC\_Channel\_21, [438](#)
- ADC\_DC\_Channel\_22, [438](#)
- ADC\_DC\_Channel\_23, [438](#)
- ADC\_DC\_Channel\_3, [437](#)
- ADC\_DC\_Channel\_4, [437](#)
- ADC\_DC\_Channel\_5, [437](#)
- ADC\_DC\_Channel\_6, [437](#)
- ADC\_DC\_Channel\_7, [437](#)
- ADC\_DC\_Channel\_8, [437](#)
- ADC\_DC\_Channel\_9, [438](#)
- ADC\_DC\_Channel\_None, [438](#)
- ADC\_DC\_Channel\_TypeDef, [437](#)
- ADC\_DC\_Cmd, [442](#)
- ADC\_DC\_Condition\_High, [438](#)
- ADC\_DC\_Condition\_Low, [438](#)
- ADC\_DC\_Condition\_TypeDef, [438](#)
- ADC\_DC\_Condition\_Window, [438](#)
- ADC\_DC\_DeInit, [442](#)
- ADC\_DC\_GetLastData, [442](#)
- ADC\_DC\_ITCmd, [444](#)
- ADC\_DC\_ITConfig, [444](#)
- ADC\_DC\_ITGenCmd, [444](#)
- ADC\_DC\_ITMaskCmd, [446](#)
- ADC\_DC\_ITMaskedStatus, [446](#)
- ADC\_DC\_ITRawStatus, [446](#)
- ADC\_DC\_ITStatusClear, [447](#)
- ADC\_DC\_Init, [442](#)
- ADC\_DC\_Mode\_Multiple, [438](#)
- ADC\_DC\_Mode\_MultipleHyst, [438](#)
- ADC\_DC\_Mode\_Single, [438](#)
- ADC\_DC\_Mode\_SingleHyst, [438](#)
- ADC\_DC\_Mode\_TypeDef, [438](#)
- ADC\_DC\_Module\_0, [439](#)
- ADC\_DC\_Module\_1, [439](#)
- ADC\_DC\_Module\_10, [439](#)
- ADC\_DC\_Module\_11, [439](#)
- ADC\_DC\_Module\_12, [439](#)
- ADC\_DC\_Module\_13, [439](#)
- ADC\_DC\_Module\_14, [439](#)
- ADC\_DC\_Module\_15, [439](#)
- ADC\_DC\_Module\_16, [439](#)
- ADC\_DC\_Module\_17, [439](#)
- ADC\_DC\_Module\_18, [439](#)
- ADC\_DC\_Module\_19, [439](#)
- ADC\_DC\_Module\_2, [439](#)
- ADC\_DC\_Module\_20, [439](#)
- ADC\_DC\_Module\_21, [439](#)
- ADC\_DC\_Module\_22, [439](#)
- ADC\_DC\_Module\_23, [439](#)
- ADC\_DC\_Module\_3, [439](#)
- ADC\_DC\_Module\_4, [439](#)
- ADC\_DC\_Module\_5, [439](#)
- ADC\_DC\_Module\_6, [439](#)
- ADC\_DC\_Module\_7, [439](#)
- ADC\_DC\_Module\_8, [439](#)
- ADC\_DC\_Module\_9, [439](#)
- ADC\_DC\_Module\_TypeDef, [438](#)
- ADC\_DC\_None, [432](#)
- ADC\_DC\_StructInit, [447](#)
- ADC\_DC\_TrigStatus, [447](#)
- ADC\_DC\_TrigStatusClear, [447](#)
- ADC\_DeInit, [448](#)
- ADC\_Init, [448](#)
- ADC\_Measure\_Diff, [439](#)
- ADC\_Measure\_Single, [439](#)
- ADC\_Measure\_TypeDef, [439](#)
- ADC\_Mode\_Active, [439](#)
- ADC\_Mode\_Powerdown, [439](#)
- ADC\_Mode\_StandBy, [439](#)
- ADC\_Mode\_TypeDef, [439](#)
- ADC\_Module\_0, [440](#)
- ADC\_Module\_1, [440](#)
- ADC\_Module\_10, [440](#)
- ADC\_Module\_11, [440](#)
- ADC\_Module\_2, [440](#)
- ADC\_Module\_3, [440](#)
- ADC\_Module\_4, [440](#)
- ADC\_Module\_5, [440](#)
- ADC\_Module\_6, [440](#)
- ADC\_Module\_7, [440](#)
- ADC\_Module\_8, [440](#)
- ADC\_Module\_9, [440](#)
- ADC\_Module\_TypeDef, [439](#)
- ADC\_Resolution\_10bit, [440](#)
- ADC\_Resolution\_12bit, [440](#)
- ADC\_Resolution\_TypeDef, [440](#)
- ADC\_SEQ\_0, [432](#)
- ADC\_SEQ\_1, [432](#)
- ADC\_SEQ\_2, [432](#)
- ADC\_SEQ\_3, [432](#)
- ADC\_SEQ\_4, [433](#)
- ADC\_SEQ\_5, [433](#)
- ADC\_SEQ\_6, [433](#)
- ADC\_SEQ\_7, [433](#)
- ADC\_SEQ\_Cmd, [448](#)
- ADC\_SEQ\_DMAMCmd, [449](#)
- ADC\_SEQ\_DMAConfig, [449](#)
- ADC\_SEQ\_DMAErrorStatus, [450](#)
- ADC\_SEQ\_DMAErrorStatusClear, [450](#)
- ADC\_SEQ\_DeInit, [449](#)
- ADC\_SEQ\_FIFOEmptyStatus, [450](#)
- ADC\_SEQ\_FIFOEmptyStatusClear, [451](#)
- ADC\_SEQ\_FIFOFullStatus, [451](#)
- ADC\_SEQ\_FIFOFullStatusClear, [451](#)
- ADC\_SEQ\_FIFOLevel\_1, [440](#)

- ADC\_SEQ\_FIFOLevel\_16, [440](#)
- ADC\_SEQ\_FIFOLevel\_2, [440](#)
- ADC\_SEQ\_FIFOLevel\_32, [440](#)
- ADC\_SEQ\_FIFOLevel\_4, [440](#)
- ADC\_SEQ\_FIFOLevel\_8, [440](#)
- ADC\_SEQ\_FIFOLevel\_TypeDef, [440](#)
- ADC\_SEQ\_GetConversionCount, [451](#)
- ADC\_SEQ\_GetFIFOData, [453](#)
- ADC\_SEQ\_GetFIFOLoad, [453](#)
- ADC\_SEQ\_GetITCount, [453](#)
- ADC\_SEQ\_ITCmd, [454](#)
- ADC\_SEQ\_ITConfig, [454](#)
- ADC\_SEQ\_ITCountRst, [455](#)
- ADC\_SEQ\_ITMaskedStatus, [455](#)
- ADC\_SEQ\_ITRawStatus, [455](#)
- ADC\_SEQ\_ITStatusClear, [456](#)
- ADC\_SEQ\_Init, [454](#)
- ADC\_SEQ\_Module\_0, [441](#)
- ADC\_SEQ\_Module\_1, [441](#)
- ADC\_SEQ\_Module\_2, [441](#)
- ADC\_SEQ\_Module\_3, [441](#)
- ADC\_SEQ\_Module\_4, [441](#)
- ADC\_SEQ\_Module\_5, [441](#)
- ADC\_SEQ\_Module\_6, [441](#)
- ADC\_SEQ\_Module\_7, [441](#)
- ADC\_SEQ\_Module\_TypeDef, [440](#)
- ADC\_SEQ\_SWReq, [456](#)
- ADC\_SEQ\_StartEvent\_CMP0, [441](#)
- ADC\_SEQ\_StartEvent\_CMP1, [441](#)
- ADC\_SEQ\_StartEvent\_CMP2, [441](#)
- ADC\_SEQ\_StartEvent\_Cycle, [441](#)
- ADC\_SEQ\_StartEvent\_ITGPIO, [441](#)
- ADC\_SEQ\_StartEvent\_PWM0, [441](#)
- ADC\_SEQ\_StartEvent\_PWM1, [441](#)
- ADC\_SEQ\_StartEvent\_PWM2, [441](#)
- ADC\_SEQ\_StartEvent\_PWM3, [441](#)
- ADC\_SEQ\_StartEvent\_PWM4, [441](#)
- ADC\_SEQ\_StartEvent\_PWM5, [441](#)
- ADC\_SEQ\_StartEvent\_SWReq, [441](#)
- ADC\_SEQ\_StartEvent\_TIM, [441](#)
- ADC\_SEQ\_StartEvent\_TypeDef, [441](#)
- ADC\_SEQ\_StructInit, [456](#)
- ADC\_StructInit, [456](#)
- IS\_ADC\_AVERAGE, [433](#)
- IS\_ADC\_DC\_CHANNEL, [433](#)
- IS\_ADC\_DC\_CONDITION, [434](#)
- IS\_ADC\_DC\_MODE, [434](#)
- IS\_ADC\_DC\_MODULE, [434](#)
- IS\_ADC\_MEASURE, [435](#)
- IS\_ADC\_MODE, [435](#)
- IS\_ADC\_MODULE, [435](#)
- IS\_ADC\_RESOLUTION, [435](#)
- IS\_ADC\_SEQ\_FIFO\_LEVEL, [436](#)
- IS\_ADC\_SEQ\_MODULE, [436](#)
- IS\_ADC\_SEQ\_START\_EVENT, [436](#)
- niietcm4\_bootflash.c, [458](#)
- BOOTFLASH\_FullErase, [459](#)
- BOOTFLASH\_ITCmd, [460](#)
- BOOTFLASH\_Info\_PageErase, [459](#)
- BOOTFLASH\_Info\_Write, [459](#)
- BOOTFLASH\_Init, [460](#)
- BOOTFLASH\_OperationStatus, [460](#)
- BOOTFLASH\_OperationStatusClear, [460](#)
- BOOTFLASH\_PageErase, [461](#)
- BOOTFLASH\_Write, [461](#)
- niietcm4\_bootflash.h, [461](#)
- BOOTFLASH\_FullErase, [464](#)
- BOOTFLASH\_INFO\_PAGE\_SIZE\_BYTES, [463](#)
- BOOTFLASH\_INFO\_PAGE\_TOTAL, [463](#)
- BOOTFLASH\_INFO\_TOTAL\_BYTES, [463](#)
- BOOTFLASH\_ITCmd, [465](#)
- BOOTFLASH\_Info\_PageErase, [464](#)
- BOOTFLASH\_Info\_Write, [465](#)
- BOOTFLASH\_Init, [465](#)
- BOOTFLASH\_OperationStatus, [465](#)
- BOOTFLASH\_OperationStatusClear, [466](#)
- BOOTFLASH\_PAGE\_SIZE\_BYTES, [463](#)
- BOOTFLASH\_PAGE\_TOTAL, [463](#)
- BOOTFLASH\_PageErase, [466](#)
- BOOTFLASH\_Status\_Complete, [464](#)
- BOOTFLASH\_Status\_Error, [464](#)
- BOOTFLASH\_Status\_None, [464](#)
- BOOTFLASH\_Status\_TypeDef, [464](#)
- BOOTFLASH\_TOTAL\_BYTES, [464](#)
- BOOTFLASH\_Write, [466](#)
- IS\_BOOTFLASH\_STATUS, [464](#)
- niietcm4\_cap.c, [466](#)
- CAP\_Capture\_Cmd, [469](#)
- CAP\_Capture\_GetCap0, [469](#)
- CAP\_Capture\_GetCap1, [469](#)
- CAP\_Capture\_GetCap2, [470](#)
- CAP\_Capture\_GetCap3, [470](#)
- CAP\_Capture\_Init, [470](#)
- CAP\_Capture\_SetCap0, [471](#)
- CAP\_Capture\_SetCap1, [471](#)
- CAP\_Capture\_SetCap2, [471](#)
- CAP\_Capture\_SetCap3, [471](#)
- CAP\_Capture\_StructInit, [472](#)
- CAP\_DeInit, [472](#)
- CAP\_GetShadowTimer, [472](#)
- CAP\_GetTimer, [473](#)
- CAP\_ITCmd, [473](#)
- CAP\_ITForceCmd, [474](#)
- CAP\_ITPendClear, [474](#)
- CAP\_ITPendStatus, [474](#)
- CAP\_ITStatus, [474](#)
- CAP\_ITStatusClear, [475](#)
- CAP\_Init, [473](#)
- CAP\_PWM\_GetCompare, [475](#)
- CAP\_PWM\_GetPeriod, [475](#)
- CAP\_PWM\_GetShadowCompare, [475](#)
- CAP\_PWM\_GetShadowPeriod, [476](#)
- CAP\_PWM\_Init, [476](#)
- CAP\_PWM\_SetCompare, [476](#)
- CAP\_PWM\_SetPeriod, [477](#)

- CAP\_PWM\_SetShadowCompare, 477
- CAP\_PWM\_SetShadowPeriod, 477
- CAP\_PWM\_StructInit, 477
- CAP\_SetShadowTimer, 479
- CAP\_SetTimer, 479
- CAP\_StructInit, 479
- CAP\_SwSync, 480
- CAP\_SyncCmd, 480
- CAP\_TimerCmd, 480
- niietcm4\_cap.h, 480
  - CAP\_Capture\_Cmd, 487
  - CAP\_Capture\_GetCap0, 488
  - CAP\_Capture\_GetCap1, 488
  - CAP\_Capture\_GetCap2, 488
  - CAP\_Capture\_GetCap3, 488
  - CAP\_Capture\_Init, 489
  - 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, 489
  - CAP\_Capture\_SetCap1, 489
  - CAP\_Capture\_SetCap2, 490
  - CAP\_Capture\_SetCap3, 490
  - CAP\_Capture\_StructInit, 490
  - CAP\_DeInit, 491
  - CAP\_GetShadowTimer, 491
  - CAP\_GetTimer, 491
  - CAP\_Halt\_Free, 487
  - CAP\_Halt\_Stop, 487
  - CAP\_Halt\_StopOnZero, 487
  - CAP\_Halt\_TypeDef, 487
  - CAP\_ITCmd, 492
  - CAP\_ITForceCmd, 492
  - CAP\_ITPendClear, 492
  - CAP\_ITPendStatus, 493
  - CAP\_ITSource\_All, 484
  - CAP\_ITSource\_CapEvent0, 484
  - CAP\_ITSource\_CapEvent1, 484
  - CAP\_ITSource\_CapEvent2, 484
  - CAP\_ITSource\_CapEvent3, 484
  - CAP\_ITSource\_GeneralInt, 484
  - CAP\_ITSource\_TimerEqCompare, 484
  - CAP\_ITSource\_TimerEqPeriod, 485
  - CAP\_ITSource\_TimerOvf, 485
  - CAP\_ITStatus, 493
  - CAP\_ITStatusClear, 493
  - CAP\_Init, 491
  - CAP\_Mode\_Capture, 487
  - CAP\_Mode\_PWM, 487
  - CAP\_Mode\_TypeDef, 487
  - CAP\_PWM\_GetCompare, 493
  - CAP\_PWM\_GetPeriod, 494
  - CAP\_PWM\_GetShadowCompare, 494
  - CAP\_PWM\_GetShadowPeriod, 494
  - CAP\_PWM\_Init, 494
  - CAP\_PWM\_Polarity\_Neg, 487
  - CAP\_PWM\_Polarity\_Pos, 487
  - CAP\_PWM\_Polarity\_TypeDef, 487
  - CAP\_PWM\_SetCompare, 496
  - CAP\_PWM\_SetPeriod, 496
  - CAP\_PWM\_SetShadowCompare, 496
  - CAP\_PWM\_SetShadowPeriod, 497
  - CAP\_PWM\_StructInit, 497
  - CAP\_SetShadowTimer, 497
  - CAP\_SetTimer, 497
  - CAP\_StructInit, 498
  - CAP\_SwSync, 498
  - CAP\_SyncCmd, 498
  - CAP\_SyncOut\_Bypass, 487
  - CAP\_SyncOut\_Disable, 487
  - CAP\_SyncOut\_TimerEqPeriod, 487
  - CAP\_SyncOut\_TypeDef, 487
  - CAP\_TimerCmd, 498
  - IS\_CAP\_CAPTURE\_MODE, 485
  - IS\_CAP\_CAPTURE\_POLARITY, 485
  - IS\_CAP\_HALT, 485
  - IS\_CAP\_MODE, 485
  - IS\_CAP\_PWM\_POLARITY, 486
  - IS\_CAP\_SYNC\_OUT, 486
- niietcm4\_conf.h, 499
- niietcm4\_dma.c, 500
  - DMA\_BasePtrConfig, 501
  - DMA\_ChannelDeInit, 502
  - DMA\_ChannelEnableCmd, 502
  - DMA\_ChannelInit, 502
  - DMA\_ChannelStructInit, 503
  - DMA\_ClearErrorStatus, 503
  - DMA\_DeInit, 503
  - DMA\_ErrorStatus, 505
  - DMA\_HighPriorityCmd, 505
  - DMA\_Init, 505
  - DMA\_MasterEnableCmd, 506
  - DMA\_MasterEnableStatus, 506
  - DMA\_PrmAltCmd, 506
  - DMA\_ProtectionConfig, 506
  - DMA\_ReqMaskCmd, 507
  - DMA\_SWRequestCmd, 508
  - DMA\_StateStatus, 507
  - DMA\_StructInit, 507
  - DMA\_UseBurstCmd, 508
  - DMA\_WaitOnReqStatus, 508
- niietcm4\_dma.h, 509
  - CHANNEL\_CFG\_CYCLE\_CTRL\_Msk, 513
  - CHANNEL\_CFG\_CYCLE\_CTRL\_Pos, 513
  - CHANNEL\_CFG\_DST\_INC\_Msk, 513
  - CHANNEL\_CFG\_DST\_INC\_Pos, 513
  - CHANNEL\_CFG\_DST\_PROT\_CTRL\_↵ Msk, 513
  - CHANNEL\_CFG\_DST\_PROT\_CTRL\_↵ Pos, 513
  - CHANNEL\_CFG\_DST\_SIZE\_Msk, 513
  - CHANNEL\_CFG\_DST\_SIZE\_Pos, 513



- CHANNEL\_CFG\_N\_MINUS\_1\_Msk, 513
- CHANNEL\_CFG\_N\_MINUS\_1\_Pos, 514
- CHANNEL\_CFG\_NEXT\_USEBURST\_ ←  
Msk, 514
- CHANNEL\_CFG\_NEXT\_USEBURST\_ ←  
Pos, 514
- CHANNEL\_CFG\_R\_POWER\_Msk, 514
- CHANNEL\_CFG\_R\_POWER\_Pos, 514
- CHANNEL\_CFG\_SRC\_INC\_Msk, 514
- CHANNEL\_CFG\_SRC\_INC\_Pos, 514
- CHANNEL\_CFG\_SRC\_PROT\_CTRL\_ ←  
Msk, 514
- CHANNEL\_CFG\_SRC\_PROT\_CTRL\_ ←  
Pos, 514
- CHANNEL\_CFG\_SRC\_SIZE\_Msk, 515
- CHANNEL\_CFG\_SRC\_SIZE\_Pos, 515
- DMA\_ArbitrationRate\_1, 522
- DMA\_ArbitrationRate\_1024, 522
- DMA\_ArbitrationRate\_128, 522
- DMA\_ArbitrationRate\_16, 522
- DMA\_ArbitrationRate\_2, 522
- DMA\_ArbitrationRate\_256, 522
- DMA\_ArbitrationRate\_32, 522
- DMA\_ArbitrationRate\_4, 522
- DMA\_ArbitrationRate\_512, 522
- DMA\_ArbitrationRate\_64, 522
- DMA\_ArbitrationRate\_8, 522
- DMA\_ArbitrationRate\_TypeDef, 522
- DMA\_BasePtrConfig, 524
- DMA\_Channel\_0, 515
- DMA\_Channel\_1, 515
- DMA\_Channel\_10, 515
- DMA\_Channel\_11, 515
- DMA\_Channel\_12, 515
- DMA\_Channel\_13, 515
- DMA\_Channel\_14, 515
- DMA\_Channel\_15, 516
- DMA\_Channel\_16, 516
- DMA\_Channel\_17, 516
- DMA\_Channel\_18, 516
- DMA\_Channel\_19, 516
- DMA\_Channel\_2, 516
- DMA\_Channel\_20, 516
- DMA\_Channel\_21, 516
- DMA\_Channel\_22, 516
- DMA\_Channel\_23, 516
- DMA\_Channel\_3, 517
- DMA\_Channel\_4, 517
- DMA\_Channel\_5, 517
- DMA\_Channel\_6, 517
- DMA\_Channel\_7, 517
- DMA\_Channel\_8, 517
- DMA\_Channel\_9, 517
- DMA\_Channel\_ADCSEQ0, 517
- DMA\_Channel\_ADCSEQ1, 517
- DMA\_Channel\_ADCSEQ2, 518
- DMA\_Channel\_ADCSEQ3, 518
- DMA\_Channel\_ADCSEQ4, 518
- DMA\_Channel\_ADCSEQ5, 518
- DMA\_Channel\_ADCSEQ6, 518
- DMA\_Channel\_ADCSEQ7, 518
- DMA\_Channel\_All, 518
- DMA\_Channel\_SPI0\_RX, 518
- DMA\_Channel\_SPI0\_TX, 518
- DMA\_Channel\_SPI1\_RX, 518
- DMA\_Channel\_SPI1\_TX, 519
- DMA\_Channel\_SPI2\_RX, 519
- DMA\_Channel\_SPI2\_TX, 519
- DMA\_Channel\_SPI3\_RX, 519
- DMA\_Channel\_SPI3\_TX, 519
- DMA\_Channel\_UART0\_RX, 519
- DMA\_Channel\_UART0\_TX, 519
- DMA\_Channel\_UART1\_RX, 519
- DMA\_Channel\_UART1\_TX, 519
- DMA\_Channel\_UART2\_RX, 520
- DMA\_Channel\_UART2\_TX, 520
- DMA\_Channel\_UART3\_RX, 520
- DMA\_Channel\_UART3\_TX, 520
- DMA\_ChannelDeInit, 525
- DMA\_ChannelEnableCmd, 525
- DMA\_ChannelInit, 525
- DMA\_ChannelStructInit, 526
- DMA\_ClearErrorStatus, 526
- DMA\_DataInc\_16, 522
- DMA\_DataInc\_32, 522
- DMA\_DataInc\_8, 522
- DMA\_DataInc\_Disable, 522
- DMA\_DataInc\_TypeDef, 522
- DMA\_DataSize\_16, 523
- DMA\_DataSize\_32, 523
- DMA\_DataSize\_8, 523
- DMA\_DataSize\_TypeDef, 522
- DMA\_DeInit, 526
- DMA\_ErrorStatus, 528
- DMA\_HighPriorityCmd, 528
- DMA\_Init, 528
- DMA\_MasterEnableCmd, 529
- DMA\_MasterEnableStatus, 529
- DMA\_Mode\_AltMemScatGath, 523
- DMA\_Mode\_AltPeriphScatGath, 523
- DMA\_Mode\_AutoReq, 523
- DMA\_Mode\_Basic, 523
- DMA\_Mode\_Disable, 523
- DMA\_Mode\_PingPong, 523
- DMA\_Mode\_PrmMemScatGath, 523
- DMA\_Mode\_PrmPeriphScatGath, 523
- DMA\_Mode\_TypeDef, 523
- DMA\_PrmAltCmd, 529
- DMA\_ProtectionConfig, 529
- DMA\_ReqMaskCmd, 530
- DMA\_SWRequestCmd, 531
- DMA\_State\_Done, 523
- DMA\_State\_Free, 523
- DMA\_State\_Pause, 523
- DMA\_State\_PeriphScatGath, 523
- DMA\_State\_ReadConfigData, 523

- DMA\_State\_ReadDstDataEndPtr, 523
- DMA\_State\_ReadSrcData, 523
- DMA\_State\_ReadSrcDataEndPtr, 523
- DMA\_State\_TypeDef, 523
- DMA\_State\_WaitReq, 523
- DMA\_State\_WriteConfigData, 523
- DMA\_State\_WriteDstData, 523
- DMA\_StateStatus, 530
- DMA\_StructInit, 530
- DMA\_UseBurstCmd, 531
- DMA\_WaitOnReqStatus, 531
- IS\_DMA\_ARBITRATION\_RATE, 520
- IS\_DMA\_DATA\_INC, 520
- IS\_DMA\_DATA\_SIZE, 521
- IS\_DMA\_MODE, 521
- IS\_DMA\_STATE, 521
- IS\_GET\_DMA\_CHANNEL, 521
- niietcm4\_extmem.c, 532
  - EXT\_MEM\_CFG\_Reset\_Value, 533
  - EXTMEM\_DeInit, 533
  - EXTMEM\_Init, 533
  - EXTMEM\_StructInit, 533
- niietcm4\_extmem.h, 534
  - EXTMEM\_CEMask\_Addr\_11, 535
  - EXTMEM\_CEMask\_Addr\_11\_19, 535
  - EXTMEM\_CEMask\_Addr\_12, 536
  - EXTMEM\_CEMask\_Addr\_13, 536
  - EXTMEM\_CEMask\_Addr\_14, 536
  - EXTMEM\_CEMask\_Addr\_15, 536
  - EXTMEM\_CEMask\_Addr\_16, 536
  - EXTMEM\_CEMask\_Addr\_17, 536
  - EXTMEM\_CEMask\_Addr\_18, 536
  - EXTMEM\_CEMask\_Addr\_19, 536
  - EXTMEM\_DeInit, 539
  - 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, 538
  - EXTMEM\_ReadWaitState\_2, 538
  - EXTMEM\_ReadWaitState\_3, 538
  - EXTMEM\_ReadWaitState\_4, 538
  - EXTMEM\_ReadWaitState\_5, 538
  - EXTMEM\_ReadWaitState\_6, 538
  - EXTMEM\_ReadWaitState\_7, 538
  - EXTMEM\_ReadWaitState\_8, 538
  - EXTMEM\_ReadWaitState\_TypeDef, 538
  - EXTMEM\_StructInit, 539
  - EXTMEM\_Width\_16bit, 538
  - EXTMEM\_Width\_8bit, 538
  - EXTMEM\_Width\_TypeDef, 538
  - EXTMEM\_WriteWaitState\_1, 539
- EXTMEM\_WriteWaitState\_2, 539
- EXTMEM\_WriteWaitState\_3, 539
- EXTMEM\_WriteWaitState\_4, 539
- EXTMEM\_WriteWaitState\_5, 539
- EXTMEM\_WriteWaitState\_6, 539
- EXTMEM\_WriteWaitState\_7, 539
- EXTMEM\_WriteWaitState\_8, 539
- EXTMEM\_WriteWaitState\_TypeDef, 538
- IS\_EXTMEM\_READ\_WAITSTATE, 536
- IS\_EXTMEM\_RW\_WAITSTATE, 537
- IS\_EXTMEM\_WIDTH, 537
- IS\_EXTMEM\_WRITE\_WAITSTATE, 537
- niietcm4\_gpio.c, 540
  - GPIO\_AltFuncConfig, 545
  - GPIO\_ClearBits, 545
  - GPIO\_DATAOUT\_Reset\_Value, 542
  - GPIO\_DeInit, 545
  - GPIO\_GPIODEN0\_Reset\_Value, 542
  - GPIO\_GPIODEN1\_Reset\_Value, 542
  - GPIO\_GPIODEN2\_Reset\_Value, 542
  - GPIO\_GPIODEN3\_Reset\_Value, 542
  - GPIO\_GPIOODCTLx\_Reset\_Value, 542
  - GPIO\_GPIOODSCTLx\_Reset\_Value, 543
  - GPIO\_GPIOPCTLx\_Reset\_Value, 543
  - GPIO\_GPIOPUCTLx\_Reset\_Value, 543
  - GPIO\_GPIOQEx\_Reset\_Value, 543
  - GPIO\_GPIOQMx\_Reset\_Value, 543
  - GPIO\_GPIOQPx\_Reset\_Value, 543
  - GPIO\_GPIOSEx\_Reset\_Value, 543
  - GPIO\_ITCmd, 546
  - GPIO\_ITConfig, 546
  - GPIO\_ITStatusClear, 547
  - GPIO\_Init, 546
  - GPIO\_QualCmd, 547
  - GPIO\_QualConfig, 547
  - GPIO\_Read, 548
  - GPIO\_ReadBit, 548
  - GPIO\_ReadMask, 548
  - GPIO\_Regs\_A\_C\_E\_G\_Mask, 543
  - GPIO\_Regs\_B\_D\_F\_H\_Mask, 544
  - GPIO\_Regs\_GPIOA\_Mask, 544
  - GPIO\_Regs\_GPIOB\_Mask, 544
  - GPIO\_Regs\_GPIOC\_Mask, 544
  - 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, 551
- 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\_AltFuncConfig, 563
- GPIO\_ClearBits, 564
- GPIO\_DeInit, 564
- GPIO\_Dir\_In, 560
- GPIO\_Dir\_Out, 560
- GPIO\_Dir\_TypeDef, 560
- GPIO\_ITCmd, 565
- GPIO\_ITConfig, 565
- GPIO\_ITStatusClear, 566
- GPIO\_Init, 564
- GPIO\_IntPol\_Neg, 561
- GPIO\_IntPol\_Pos, 561
- GPIO\_IntPol\_TypeDef, 560
- GPIO\_IntType\_Edge, 561
- GPIO\_IntType\_Level, 561
- GPIO\_IntType\_TypeDef, 561
- GPIO\_Load\_16mA, 561
- GPIO\_Load\_8mA, 561
- GPIO\_Load\_TypeDef, 561
- 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, 562
- GPIO\_OutMode\_PP, 562
- GPIO\_OutMode\_TypeDef, 561
- GPIO\_Pin\_0, 554
- GPIO\_Pin\_0\_3, 555
- GPIO\_Pin\_0\_7, 555
- 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, 556
- GPIO\_Pin\_2, 556
- 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, 557
- GPIO\_Pin\_8\_15, 557
- GPIO\_Pin\_9, 557
- GPIO\_Pin\_All, 557
- GPIO\_PullUp\_Dis, 562
- GPIO\_PullUp\_En, 562
- GPIO\_PullUp\_TypeDef, 562
- GPIO\_Qual\_Dis, 562
- GPIO\_Qual\_En, 562
- GPIO\_Qual\_TypeDef, 562
- GPIO\_QualCmd, 566
- GPIO\_QualConfig, 566
- GPIO\_QualMode\_3sample, 562
- GPIO\_QualMode\_6sample, 562
- GPIO\_QualMode\_TypeDef, 562
- GPIO\_Read, 567
- GPIO\_ReadBit, 567
- GPIO\_ReadMask, 567
- GPIO\_SetBits, 568
- GPIO\_StructInit, 568
- GPIO\_Sync\_Dis, 562
- GPIO\_Sync\_En, 562
- GPIO\_Sync\_TypeDef, 562
- GPIO\_SyncCmd, 568
- GPIO\_ToggleBits, 569
- GPIO\_Write, 569
- GPIO\_WriteBit, 569
- GPIO\_WriteMask, 570
- IS\_GET\_GPIO\_PIN, 557
- IS\_GPIO\_ALT\_FUNC, 557
- IS\_GPIO\_DIR, 558
- IS\_GPIO\_INT\_POL, 558
- IS\_GPIO\_INT\_TYPE, 558
- IS\_GPIO\_LOAD, 558
- IS\_GPIO\_MODE, 558
- IS\_GPIO\_OUT, 559
- IS\_GPIO\_OUT\_MODE, 559
- IS\_GPIO\_PULLUP, 559
- IS\_GPIO\_QUAL, 559
- IS\_GPIO\_QUAL\_MODE, 559
- IS\_GPIO\_SYNC, 560
- niitcm4\_rcc.c, 570
- RCC\_ADCClkCmd, 572
- RCC\_ADCClkDivConfig, 573
- RCC\_PLL\_CTRL\_Reset\_Value, 572
- RCC\_PLL\_NF\_Reset\_Value, 572
- RCC\_PLL\_NR\_Reset\_Value, 572
- RCC\_PLL\_OD\_Reset\_Value, 572
- RCC\_PLLAutoConfig, 574
- RCC\_PLLDeInit, 575
- RCC\_PLLInit, 575
- RCC\_PLLPowerDownCmd, 576
- RCC\_PLLStructInit, 576
- RCC\_PeriphClkCmd, 573
- RCC\_PeriphRstCmd, 574
- RCC\_SPIClkCmd, 576
- RCC\_SPIClkDivConfig, 577
- RCC\_SPIClkSel, 577
- RCC\_SysClkDiv2Out, 577
- RCC\_SysClkSel, 578
- RCC\_SysClkStatus, 578
- RCC\_UARTClkCmd, 578
- RCC\_UARTClkDivConfig, 579
- RCC\_UARTClkSel, 579

- RCC\_USBCLKCmd, 579
- RCC\_USBCLKConfig, 580
- RCC\_WaitClkChange, 580
- niietcm4\_rcc.h, 580
- IS\_RCC\_ADC\_CLK, 584
- IS\_RCC\_PERIPH\_CLK, 584
- IS\_RCC\_PLL\_NO, 585
- IS\_RCC\_PLL\_REF, 585
- IS\_RCC\_SPI\_CLK, 585
- IS\_RCC\_SYS\_CLK, 585
- IS\_RCC\_UART\_CLK, 585
- IS\_RCC\_USB\_CLK, 586
- IS\_RCC\_USB\_FREQ, 586
- RCC\_ADCClk\_0, 587
- RCC\_ADCClk\_1, 587
- RCC\_ADCClk\_10, 587
- RCC\_ADCClk\_11, 587
- RCC\_ADCClk\_2, 587
- RCC\_ADCClk\_3, 587
- RCC\_ADCClk\_4, 587
- RCC\_ADCClk\_5, 587
- RCC\_ADCClk\_6, 587
- RCC\_ADCClk\_7, 587
- RCC\_ADCClk\_8, 587
- RCC\_ADCClk\_9, 587
- RCC\_ADCClk\_TypeDef, 586
- RCC\_ADCClkCmd, 590
- RCC\_ADCClkDivConfig, 590
- RCC\_CLK\_CHANGE\_TIMEOUT, 586
- RCC\_CLK\_PLL\_STABLE\_TIMEOUT, 586
- RCC\_PLLAutoConfig, 592
- RCC\_PLLDeInit, 592
- RCC\_PLLInit, 592
- RCC\_PLLNO\_Disable, 588
- RCC\_PLLNO\_Div2, 588
- RCC\_PLLNO\_Div4, 588
- RCC\_PLLNO\_TypeDef, 588
- RCC\_PLLPowerDownCmd, 593
- RCC\_PLLRef\_ETH\_25MHz, 589
- RCC\_PLLRef\_TypeDef, 588
- RCC\_PLLRef\_USB\_60MHz, 589
- RCC\_PLLRef\_USB\_CLK, 589
- RCC\_PLLRef\_XI\_OSC, 589
- RCC\_PLLStructInit, 594
- RCC\_PeriphClk\_ADC, 587
- RCC\_PeriphClk\_CMP, 587
- RCC\_PeriphClk\_I2C0, 587
- RCC\_PeriphClk\_I2C1, 587
- RCC\_PeriphClk\_PWM0, 587
- RCC\_PeriphClk\_PWM1, 587
- RCC\_PeriphClk\_PWM2, 587
- RCC\_PeriphClk\_PWM3, 587
- RCC\_PeriphClk\_PWM4, 587
- RCC\_PeriphClk\_PWM5, 587
- RCC\_PeriphClk\_PWM6, 587
- RCC\_PeriphClk\_PWM7, 587
- RCC\_PeriphClk\_PWM8, 587
- RCC\_PeriphClk\_QEP0, 587
- RCC\_PeriphClk\_QEP1, 587
- RCC\_PeriphClk\_TypeDef, 587
- RCC\_PeriphClk\_WD, 587
- RCC\_PeriphClkCmd, 591
- RCC\_PeriphRst\_CAP0, 588
- RCC\_PeriphRst\_CAP1, 588
- RCC\_PeriphRst\_CAP2, 588
- RCC\_PeriphRst\_CAP3, 588
- RCC\_PeriphRst\_CAP4, 588
- RCC\_PeriphRst\_CAP5, 588
- RCC\_PeriphRst\_CMP, 588
- RCC\_PeriphRst\_ETH, 588
- RCC\_PeriphRst\_I2C0, 587
- RCC\_PeriphRst\_I2C1, 588
- RCC\_PeriphRst\_PWM0, 588
- RCC\_PeriphRst\_PWM1, 588
- RCC\_PeriphRst\_PWM2, 588
- RCC\_PeriphRst\_PWM3, 588
- RCC\_PeriphRst\_PWM4, 588
- RCC\_PeriphRst\_PWM5, 588
- RCC\_PeriphRst\_PWM6, 588
- RCC\_PeriphRst\_PWM7, 588
- RCC\_PeriphRst\_PWM8, 588
- RCC\_PeriphRst\_QEP0, 588
- RCC\_PeriphRst\_QEP1, 588
- RCC\_PeriphRst\_SPI0, 588
- RCC\_PeriphRst\_SPI1, 588
- RCC\_PeriphRst\_SPI2, 588
- RCC\_PeriphRst\_SPI3, 588
- RCC\_PeriphRst\_Timer0, 588
- RCC\_PeriphRst\_Timer1, 588
- RCC\_PeriphRst\_Timer2, 588
- RCC\_PeriphRst\_TypeDef, 587
- RCC\_PeriphRst\_UART0, 588
- RCC\_PeriphRst\_UART1, 588
- RCC\_PeriphRst\_UART2, 588
- RCC\_PeriphRst\_UART3, 588
- RCC\_PeriphRst\_USB, 588
- RCC\_PeriphRst\_WD, 587
- RCC\_PeriphRstCmd, 591
- RCC\_SPIClk\_SYSClk, 589
- RCC\_SPIClk\_TypeDef, 589
- RCC\_SPIClk\_USB\_60MHz, 589
- RCC\_SPIClk\_USB\_CLK, 589
- RCC\_SPIClk\_XI\_OSC, 589
- RCC\_SPIClkCmd, 594
- RCC\_SPIClkDivConfig, 594
- RCC\_SPIClkSel, 595
- RCC\_SysClk\_CPE\_Sel, 589
- RCC\_SysClk\_ETH25MHz, 589
- RCC\_SysClk\_PLL, 589
- RCC\_SysClk\_PLLDIV, 589
- RCC\_SysClk\_POR, 589
- RCC\_SysClk\_TypeDef, 589
- RCC\_SysClk\_USB60MHz, 589
- RCC\_SysClk\_USB\_CLK, 589
- RCC\_SysClk\_XI\_OSC, 589

- RCC\_SysClkDiv2Out, 595
- RCC\_SysClkSel, 595
- RCC\_SysClkStatus, 596
- RCC\_UARTClk\_SYSCLK, 589
- RCC\_UARTClk\_TypeDef, 589
- RCC\_UARTClk\_USB\_60MHz, 589
- RCC\_UARTClk\_USB\_CLK, 589
- RCC\_UARTClk\_XI\_OSC, 589
- RCC\_UARTClkCmd, 596
- RCC\_UARTClkDivConfig, 596
- RCC\_UARTClkSel, 596
- RCC\_USBClk\_TypeDef, 589
- RCC\_USBClk\_USB\_CLK, 590
- RCC\_USBClk\_XI\_OSC, 590
- RCC\_USBClkCmd, 597
- RCC\_USBClkConfig, 597
- RCC\_USBFreq\_12MHz, 590
- RCC\_USBFreq\_24MHz, 590
- RCC\_USBFreq\_TypeDef, 590
- niietcm4\_rtc.c, 597
- niietcm4\_rtc.h, 598
  - IS\_RTC\_FORMAT, 600
  - IS\_RTC\_MONTH, 600
  - IS\_RTC\_WEEKDAY, 601
  - RTC\_Format\_BCD, 601
  - RTC\_Format\_BIN, 601
  - RTC\_Format\_TypeDef, 601
  - RTC\_Month\_April, 601
  - RTC\_Month\_August, 601
  - RTC\_Month\_December, 601
  - RTC\_Month\_February, 601
  - RTC\_Month\_January, 601
  - RTC\_Month\_July, 601
  - RTC\_Month\_June, 601
  - RTC\_Month\_March, 601
  - RTC\_Month\_May, 601
  - RTC\_Month\_November, 601
  - RTC\_Month\_October, 601
  - RTC\_Month\_September, 601
  - RTC\_Month\_TypeDef, 601
  - RTC\_Weekday\_Friday, 602
  - RTC\_Weekday\_Monday, 602
  - RTC\_Weekday\_Saturday, 602
  - RTC\_Weekday\_Sunday, 602
  - RTC\_Weekday\_Thursday, 602
  - RTC\_Weekday\_Tuesday, 602
  - RTC\_Weekday\_TypeDef, 601
  - RTC\_Weekday\_Wednesday, 602
- niietcm4\_timer.c, 602
  - TIMER\_Cmd, 603
  - TIMER\_ExtInputConfig, 603
  - TIMER\_FreqConfig, 604
  - TIMER\_GetCounter, 604
  - TIMER\_GetReload, 604
  - TIMER\_ITCmd, 605
  - TIMER\_ITStatus, 605
  - TIMER\_ITStatusClear, 605
  - TIMER\_PeriodConfig, 605
  - TIMER\_SetCounter, 606
  - TIMER\_SetReload, 606
- niietcm4\_timer.h, 606
  - IS\_TIMER\_EXT\_INPUT, 608
  - TIMER\_Cmd, 609
  - TIMER\_ExtInput\_CountClk, 608
  - TIMER\_ExtInput\_CountEn, 608
  - TIMER\_ExtInput\_Disable, 608
  - TIMER\_ExtInput\_TypeDef, 608
  - TIMER\_ExtInputConfig, 610
  - TIMER\_FreqConfig, 610
  - TIMER\_GetCounter, 610
  - TIMER\_GetReload, 611
  - TIMER\_ITCmd, 611
  - TIMER\_ITStatus, 611
  - TIMER\_ITStatusClear, 611
  - TIMER\_PeriodConfig, 612
  - TIMER\_SetCounter, 612
  - TIMER\_SetReload, 612
- niietcm4\_uart.c, 613
  - UART\_BaudRateDivConfig, 615
  - UART\_Break, 616
  - UART\_Cmd, 616
  - UART\_DMABlkOnErrCmd, 617
  - UART\_DMACmd, 617
  - UART\_DeInit, 616
  - UART\_ErrorStatus, 617
  - UART\_ErrorStatusClear, 618
  - UART\_FlagStatus, 618
  - UART\_ITCmd, 619
  - UART\_ITFIFOLevelConfig, 619
  - UART\_ITMaskedStatus, 619
  - UART\_ITRawStatus, 620
  - UART\_ITStatusClear, 620
  - UART\_Init, 618
  - UART\_ModemConfig, 620
  - UART\_ModemStructInit, 621
  - UART\_RecieveData, 621
  - UART\_SendData, 621
  - UART\_StructInit, 622
- niietcm4\_uart.h, 622
  - IS\_UART\_DATA\_WIDTH, 625
  - IS\_UART\_DIR, 625
  - IS\_UART\_FIFO\_LEVEL, 626
  - IS\_UART\_PARITY\_BIT, 626
  - IS\_UART\_STOP\_BIT, 626
  - UART\_BaudRateDivConfig, 630
  - UART\_Break, 631
  - UART\_Cmd, 631
  - UART\_DMABlkOnErrCmd, 632
  - UART\_DMACmd, 632
  - UART\_DataWidth\_5, 629
  - UART\_DataWidth\_6, 629
  - UART\_DataWidth\_7, 629
  - UART\_DataWidth\_8, 629
  - UART\_DataWidth\_TypeDef, 629
  - UART\_DeInit, 631
  - UART\_Dir\_Rx, 630

- UART\_Dir\_Tx, [630](#)
- UART\_Dir\_Typedef, [629](#)
- UART\_Error\_All, [626](#)
- UART\_Error\_Break, [626](#)
- UART\_Error\_Frame, [627](#)
- UART\_Error\_Overflow, [627](#)
- UART\_Error\_Parity, [627](#)
- UART\_ErrorStatus, [632](#)
- UART\_ErrorStatusClear, [633](#)
- UART\_FIFOLevel\_1\_2, [630](#)
- UART\_FIFOLevel\_1\_4, [630](#)
- UART\_FIFOLevel\_1\_8, [630](#)
- UART\_FIFOLevel\_3\_4, [630](#)
- UART\_FIFOLevel\_7\_8, [630](#)
- UART\_FIFOLevel\_TypeDef, [630](#)
- UART\_Flag\_All, [627](#)
- UART\_Flag\_Busy, [627](#)
- UART\_Flag\_InvCTS, [627](#)
- UART\_Flag\_InvDCD, [627](#)
- UART\_Flag\_InvDSR, [627](#)
- UART\_Flag\_InvRI, [627](#)
- UART\_Flag\_RxFIFOEmpty, [628](#)
- UART\_Flag\_RxFIFOFull, [628](#)
- UART\_Flag\_TxFIFOEmpty, [628](#)
- UART\_Flag\_TxFIFOFull, [628](#)
- UART\_FlagStatus, [633](#)
- UART\_ITCmd, [634](#)
- UART\_ITFIFOLevelConfig, [634](#)
- UART\_ITMaskedStatus, [634](#)
- UART\_ITRawStatus, [635](#)
- UART\_ITSource\_All, [628](#)
- UART\_ITSource\_ChangeCTS, [628](#)
- UART\_ITSource\_ChangeDCD, [628](#)
- UART\_ITSource\_ChangeDSR, [628](#)
- UART\_ITSource\_ChangeRI, [628](#)
- UART\_ITSource\_ErrorBreak, [628](#)
- UART\_ITSource\_ErrorFrame, [629](#)
- UART\_ITSource\_ErrorOverflow, [629](#)
- UART\_ITSource\_ErrorParity, [629](#)
- UART\_ITSource\_RecieveTimeout, [629](#)
- UART\_ITSource\_RxFIFOLevel, [629](#)
- UART\_ITSource\_TxFIFOLevel, [629](#)
- UART\_ITStatusClear, [635](#)
- UART\_Init, [633](#)
- UART\_ModemConfig, [635](#)
- UART\_ModemStructInit, [636](#)
- UART\_ParityBit\_Disable, [630](#)
- UART\_ParityBit\_Even, [630](#)
- UART\_ParityBit\_High, [630](#)
- UART\_ParityBit\_Low, [630](#)
- UART\_ParityBit\_Odd, [630](#)
- UART\_ParityBit\_TypeDef, [630](#)
- UART\_RecieveData, [636](#)
- UART\_SendData, [636](#)
- UART\_StopBit\_1, [630](#)
- UART\_StopBit\_2, [630](#)
- UART\_StopBit\_TypeDef, [630](#)
- UART\_StructInit, [637](#)
- niietcm4\_userflash.c, [637](#)
  - USERFLASH\_FullErase, [638](#)
  - USERFLASH\_ITCmd, [639](#)
  - USERFLASH\_Info\_PageErase, [638](#)
  - USERFLASH\_Info\_Read, [639](#)
  - USERFLASH\_Info\_Write, [639](#)
  - USERFLASH\_Init, [639](#)
  - USERFLASH\_OperationStatus, [640](#)
  - USERFLASH\_OperationStatusClear, [640](#)
  - USERFLASH\_PageErase, [640](#)
  - USERFLASH\_Read, [640](#)
  - USERFLASH\_Write, [641](#)
- niietcm4\_userflash.h, [641](#)
  - IS\_USERFLASH\_STATUS, [643](#)
  - USERFLASH\_FullErase, [644](#)
  - USERFLASH\_INFO\_PAGE\_SIZE\_BYT↵ES, [643](#)
  - USERFLASH\_INFO\_PAGE\_TOTAL, [643](#)
  - USERFLASH\_INFO\_TOTAL\_BYTES, [643](#)
  - USERFLASH\_ITCmd, [645](#)
  - USERFLASH\_Info\_PageErase, [644](#)
  - USERFLASH\_Info\_Read, [645](#)
  - USERFLASH\_Info\_Write, [645](#)
  - USERFLASH\_Init, [645](#)
  - USERFLASH\_OperationStatus, [646](#)
  - USERFLASH\_OperationStatusClear, [646](#)
  - USERFLASH\_PAGE\_SIZE\_BYTES, [643](#)
  - USERFLASH\_PAGE\_TOTAL, [644](#)
  - USERFLASH\_PageErase, [646](#)
  - USERFLASH\_Read, [646](#)
  - USERFLASH\_Status\_Complete, [644](#)
  - USERFLASH\_Status\_Error, [644](#)
  - USERFLASH\_Status\_None, [644](#)
  - USERFLASH\_Status\_TypeDef, [644](#)
  - USERFLASH\_TOTAL\_BYTES, [644](#)
  - USERFLASH\_Write, [647](#)
- niietcm4\_watchdog.c, [647](#)
  - WATCHDOG\_Cmd, [649](#)
  - WATCHDOG\_GetCounter, [649](#)
  - WATCHDOG\_GetReload, [649](#)
  - WATCHDOG\_ITMaskedStatus, [649](#)
  - WATCHDOG\_ITRawStatus, [649](#)
  - WATCHDOG\_ITStatusClear, [649](#)
  - WATCHDOG\_Lock\_Value, [648](#)
  - WATCHDOG\_LockCmd, [650](#)
  - WATCHDOG\_RstCmd, [650](#)
  - WATCHDOG\_SetReload, [650](#)
  - WATCHDOG\_Unlock\_Value, [648](#)
- niietcm4\_watchdog.h, [650](#)
  - WATCHDOG\_Cmd, [652](#)
  - WATCHDOG\_GetCounter, [652](#)
  - WATCHDOG\_GetReload, [652](#)
  - WATCHDOG\_ITMaskedStatus, [652](#)
  - WATCHDOG\_ITRawStatus, [653](#)
  - WATCHDOG\_ITStatusClear, [653](#)
  - WATCHDOG\_LockCmd, [653](#)
  - WATCHDOG\_RstCmd, [653](#)
  - WATCHDOG\_SetReload, [653](#)

- PRIVELGED
  - DMA\_Protect\_TypeDef, [389](#)
- PRM\_DATA
  - DMA\_ConfigData\_TypeDef, [386](#)
- R\_POWER
  - \_CHANNEL\_CFG\_bits, [372](#)
- RCC, [314](#)
- RCC\_ADCClk\_0
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_1
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_10
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_11
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_2
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_3
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_4
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_5
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_6
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_7
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_8
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_9
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_ADCClk\_TypeDef
  - Типы, [166](#)
  - niietcm4\_rcc.h, [586](#)
- RCC\_ADCClkCmd
  - Приватные функции, [319](#)
  - Тактирование ADC, [182](#)
  - niietcm4\_rcc.c, [572](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_ADCClkDivConfig
  - Приватные функции, [319](#)
  - Тактирование ADC, [182](#)
  - niietcm4\_rcc.c, [573](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_CLK\_CHANGE\_TIMEOUT
  - Константы, [161](#)
  - niietcm4\_rcc.h, [586](#)
- RCC\_CLK\_PLL\_STABLE\_TIMEOUT
  - Константы, [161](#)
  - niietcm4\_rcc.h, [586](#)
- RCC\_PLL\_CTRL\_Reset\_Value
  - Начальные значения регистров, [317](#)
  - niietcm4\_rcc.c, [572](#)
- RCC\_PLL\_NF\_Reset\_Value
  - Начальные значения регистров, [317](#)
  - niietcm4\_rcc.c, [572](#)
- RCC\_PLL\_NR\_Reset\_Value
  - Начальные значения регистров, [317](#)
  - niietcm4\_rcc.c, [572](#)
- RCC\_PLL\_OD\_Reset\_Value
  - Начальные значения регистров, [317](#)
  - niietcm4\_rcc.c, [572](#)
- RCC\_PLLAutoConfig
  - Конфигурация PLL, [171](#)
  - Приватные функции, [320](#)
  - niietcm4\_rcc.c, [574](#)
  - niietcm4\_rcc.h, [592](#)
- RCC\_PLLDeInit
  - Конфигурация PLL, [171](#)
  - Приватные функции, [321](#)
  - niietcm4\_rcc.c, [575](#)
  - niietcm4\_rcc.h, [592](#)
- RCC\_PLLDiv
  - RCC\_PLLInit\_TypeDef, [392](#)
- RCC\_PLLInit
  - Конфигурация PLL, [172](#)
  - Приватные функции, [321](#)
  - niietcm4\_rcc.c, [575](#)
  - niietcm4\_rcc.h, [592](#)
- RCC\_PLLInit\_TypeDef, [392](#)
  - RCC\_PLLDiv, [392](#)
  - RCC\_PLLNF, [392](#)
  - RCC\_PLLNO, [392](#)
  - RCC\_PLLNR, [393](#)
  - RCC\_PLLRef, [393](#)
- RCC\_PLLNF
  - RCC\_PLLInit\_TypeDef, [392](#)
- RCC\_PLLNO
  - RCC\_PLLInit\_TypeDef, [392](#)
- RCC\_PLLNO\_Disable
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PLLNO\_Div2
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PLLNO\_Div4
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PLLNO\_TypeDef
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PLLNR
  - RCC\_PLLInit\_TypeDef, [393](#)
- RCC\_PLLPowerDownCmd

- Конфигурация PLL, [172](#)
- Приватные функции, [322](#)
- niietcm4\_rcc.c, [576](#)
- niietcm4\_rcc.h, [593](#)
- RCC\_PLLRef
  - RCC\_PLLInit\_TypeDef, [393](#)
- RCC\_PLLRef\_ETH\_25MHz
  - Типы, [168](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_PLLRef\_TypeDef
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PLLRef\_USB\_60MHz
  - Типы, [168](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_PLLRef\_USB\_CLK
  - Типы, [168](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_PLLRef\_XI\_OSC
  - Типы, [168](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_PLLStructInit
  - Конфигурация PLL, [173](#)
  - Приватные функции, [322](#)
  - niietcm4\_rcc.c, [576](#)
  - niietcm4\_rcc.h, [594](#)
- RCC\_PeriphClk\_ADC
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_CMP
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_I2C0
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_I2C1
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM0
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM1
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM2
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM3
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM4
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM5
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM6
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM7
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_PWM8
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_QEP0
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_QEP1
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_TypeDef
  - Типы, [166](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClk\_WD
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphClkCmd
  - Приватные функции, [319](#)
  - Управление тактированием, [174](#)
  - niietcm4\_rcc.c, [573](#)
  - niietcm4\_rcc.h, [591](#)
- RCC\_PeriphRst\_CAP0
  - Типы, [167](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_CAP1
  - Типы, [167](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_CAP2
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_CAP3
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_CAP4
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_CAP5
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_CMP
  - Типы, [168](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_ETH
  - Типы, [167](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_I2C0
  - Типы, [167](#)
  - niietcm4\_rcc.h, [587](#)
- RCC\_PeriphRst\_I2C1
  - Типы, [167](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_PWM0
  - Типы, [167](#)
  - niietcm4\_rcc.h, [588](#)
- RCC\_PeriphRst\_PWM1
  - Типы, [167](#)
  - niietcm4\_rcc.h, [588](#)



- Типы, [167](#)
- [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM2
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM3
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM4
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM5
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM6
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM7
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_PWM8
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_QEP0
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_QEP1
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_SPI0
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_SPI1
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_SPI2
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_SPI3
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_Timer0
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_Timer1
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_Timer2
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_TypeDef
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [587](#)
- RCC\_PeriphRst\_UART0
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_UART1
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_UART2
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_UART3
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_USB
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [588](#)
- RCC\_PeriphRst\_WD
  - Типы, [167](#)
  - [niietcm4\\_rcc.h](#), [587](#)
- RCC\_PeriphRstCmd
  - Приватные функции, [320](#)
  - Управление сбросом, [183](#)
  - [niietcm4\\_rcc.c](#), [574](#)
  - [niietcm4\\_rcc.h](#), [591](#)
- RCC\_SPIClk\_SYSClk
  - Типы, [168](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SPIClk\_TypeDef
  - Типы, [168](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SPIClk\_USB\_60MHz
  - Типы, [168](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SPIClk\_USB\_CLK
  - Типы, [168](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SPIClk\_XI\_OSC
  - Типы, [168](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SPIClkCmd
  - Приватные функции, [322](#)
  - Тактирование SPI, [180](#)
  - [niietcm4\\_rcc.c](#), [576](#)
  - [niietcm4\\_rcc.h](#), [594](#)
- RCC\_SPIClkDivConfig
  - Приватные функции, [323](#)
  - Тактирование SPI, [180](#)
  - [niietcm4\\_rcc.c](#), [577](#)
  - [niietcm4\\_rcc.h](#), [594](#)
- RCC\_SPIClkSel
  - Приватные функции, [323](#)
  - Тактирование SPI, [180](#)
  - [niietcm4\\_rcc.c](#), [577](#)
  - [niietcm4\\_rcc.h](#), [595](#)
- RCC\_SysClk\_CPE\_Sel
  - Типы, [169](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SysClk\_ETH25MHz
  - Типы, [169](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SysClk\_PLL
  - Типы, [169](#)
  - [niietcm4\\_rcc.h](#), [589](#)
- RCC\_SysClk\_PLLDIV

- Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_SysClk\_POR
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_SysClk\_TypeDef
  - Типы, [168](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_SysClk\_USB60MHz
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_SysClk\_USB\_CLK
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_SysClk\_XI\_OSC
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_SysClkDiv2Out
  - Функции, [170](#)
  - Приватные функции, [323](#)
  - niietcm4\_rcc.c, [577](#)
  - niietcm4\_rcc.h, [595](#)
- RCC\_SysClkSel
  - Приватные функции, [324](#)
  - Управление тактированием, [174](#)
  - niietcm4\_rcc.c, [578](#)
  - niietcm4\_rcc.h, [595](#)
- RCC\_SysClkStatus
  - Приватные функции, [324](#)
  - Управление тактированием, [175](#)
  - niietcm4\_rcc.c, [578](#)
  - niietcm4\_rcc.h, [596](#)
- RCC\_UARTClk\_SYSClk
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_UARTClk\_TypeDef
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_UARTClk\_USB\_60MHz
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_UARTClk\_USB\_CLK
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_UARTClk\_XI\_OSC
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_UARTClkCmd
  - Приватные функции, [324](#)
  - Тактирование UART, [177](#)
  - niietcm4\_rcc.c, [578](#)
  - niietcm4\_rcc.h, [596](#)
- RCC\_UARTClkDivConfig
  - Приватные функции, [325](#)
  - Тактирование UART, [177](#)
  - niietcm4\_rcc.c, [579](#)
  - niietcm4\_rcc.h, [596](#)
- RCC\_UARTClkSel
  - Приватные функции, [325](#)
  - Тактирование UART, [177](#)
  - niietcm4\_rcc.c, [579](#)
  - niietcm4\_rcc.h, [596](#)
- RCC\_USBClk\_TypeDef
  - Типы, [169](#)
  - niietcm4\_rcc.h, [589](#)
- RCC\_USBClk\_USB\_CLK
  - Типы, [169](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_USBClk\_XI\_OSC
  - Типы, [169](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_USBClkCmd
  - Приватные функции, [325](#)
  - Тактирование USB, [176](#)
  - niietcm4\_rcc.c, [579](#)
  - niietcm4\_rcc.h, [597](#)
- RCC\_USBClkConfig
  - Приватные функции, [326](#)
  - Тактирование USB, [176](#)
  - niietcm4\_rcc.c, [580](#)
  - niietcm4\_rcc.h, [597](#)
- RCC\_USBFreq\_12MHz
  - Типы, [169](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_USBFreq\_24MHz
  - Типы, [169](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_USBFreq\_TypeDef
  - Типы, [169](#)
  - niietcm4\_rcc.h, [590](#)
- RCC\_WaitClkChange
  - Приватные функции, [326](#)
  - niietcm4\_rcc.c, [580](#)
- RESERVED0
  - DMA\_ConfigData\_TypeDef, [386](#)
- RESERVED1
  - DMA\_ConfigData\_TypeDef, [386](#)
- RTC, [327](#)
- RTC\_Date\_TypeDef, [393](#)
  - RTC\_Day, [393](#)
  - RTC\_Month, [393](#)
  - RTC\_Weekday, [394](#)
  - RTC\_Year, [394](#)
- RTC\_Day
  - RTC\_Date\_TypeDef, [393](#)
- RTC\_Format\_BCD
  - Типы, [185](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Format\_BIN
  - Типы, [185](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Format\_TypeDef
  - Типы, [185](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Hour
  - RTC\_Time\_TypeDef, [394](#)



- RTC\_Minute
  - RTC\_Time\_TypeDef, [394](#)
- RTC\_Month
  - RTC\_Date\_TypeDef, [393](#)
- RTC\_Month\_April
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_August
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_December
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_February
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_January
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_July
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_June
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_March
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_May
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_November
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_October
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_September
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Month\_TypeDef
  - Типы, [185](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Psecond
  - RTC\_Time\_TypeDef, [395](#)
- RTC\_Second
  - RTC\_Time\_TypeDef, [395](#)
- RTC\_Time\_TypeDef, [394](#)
  - RTC\_Hour, [394](#)
  - RTC\_Minute, [394](#)
  - RTC\_Psecond, [395](#)
  - RTC\_Second, [395](#)
- RTC\_Weekday
  - RTC\_Date\_TypeDef, [394](#)
- RTC\_Weekday\_Friday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Weekday\_Monday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Weekday\_Saturday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Weekday\_Sunday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Weekday\_Thursday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Weekday\_Tuesday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Weekday\_TypeDef
  - Типы, [186](#)
  - niietcm4\_rtc.h, [601](#)
- RTC\_Weekday\_Wednesday
  - Типы, [186](#)
  - niietcm4\_rtc.h, [602](#)
- RTC\_Year
  - RTC\_Date\_TypeDef, [394](#)
- SRC\_DATA\_END
  - DMA\_Channel\_TypeDef, [382](#)
- SRC\_INC
  - \_CHANNEL\_CFG\_bits, [373](#)
- SRC\_PROT\_BUFFERABLE
  - \_CHANNEL\_CFG\_bits, [373](#)
- SRC\_PROT\_CACHEABLE
  - \_CHANNEL\_CFG\_bits, [373](#)
- SRC\_PROT\_PRIVILEGED
  - \_CHANNEL\_CFG\_bits, [373](#)
- SRC\_SIZE
  - \_CHANNEL\_CFG\_bits, [373](#)
- TIMER, [330](#)
- TIMER\_Cmd
  - Конфигурация, [191](#)
  - Приватные функции, [333](#)
  - niietcm4\_timer.c, [603](#)
  - niietcm4\_timer.h, [609](#)
- TIMER\_ExtInput\_CountClk
  - Типы, [188](#)
  - niietcm4\_timer.h, [608](#)
- TIMER\_ExtInput\_CountEn
  - Типы, [188](#)
  - niietcm4\_timer.h, [608](#)
- TIMER\_ExtInput\_Disable
  - Типы, [188](#)
  - niietcm4\_timer.h, [608](#)
- TIMER\_ExtInput\_TypeDef
  - Типы, [188](#)
  - niietcm4\_timer.h, [608](#)
- TIMER\_ExtInputConfig
  - Конфигурация, [191](#)
  - Приватные функции, [333](#)
  - niietcm4\_timer.c, [603](#)
  - niietcm4\_timer.h, [610](#)

- TIMER\_FreqConfig
  - Конфигурация, [192](#)
  - Приватные функции, [334](#)
  - niietcm4\_timer.c, [604](#)
  - niietcm4\_timer.h, [610](#)
- TIMER\_GetCounter
  - Конфигурация, [192](#)
  - Приватные функции, [334](#)
  - niietcm4\_timer.c, [604](#)
  - niietcm4\_timer.h, [610](#)
- TIMER\_GetReload
  - Конфигурация, [192](#)
  - Приватные функции, [334](#)
  - niietcm4\_timer.c, [604](#)
  - niietcm4\_timer.h, [611](#)
- TIMER\_ITCmd
  - Прерывания, [195](#)
  - Приватные функции, [335](#)
  - niietcm4\_timer.c, [605](#)
  - niietcm4\_timer.h, [611](#)
- TIMER\_ITStatus
  - Прерывания, [195](#)
  - Приватные функции, [335](#)
  - niietcm4\_timer.c, [605](#)
  - niietcm4\_timer.h, [611](#)
- TIMER\_ITStatusClear
  - Прерывания, [195](#)
  - Приватные функции, [335](#)
  - niietcm4\_timer.c, [605](#)
  - niietcm4\_timer.h, [611](#)
- TIMER\_PeriodConfig
  - Конфигурация, [193](#)
  - Приватные функции, [335](#)
  - niietcm4\_timer.c, [605](#)
  - niietcm4\_timer.h, [612](#)
- TIMER\_SetCounter
  - Конфигурация, [193](#)
  - Приватные функции, [336](#)
  - niietcm4\_timer.c, [606](#)
  - niietcm4\_timer.h, [612](#)
- TIMER\_SetReload
  - Конфигурация, [193](#)
  - Приватные функции, [336](#)
  - niietcm4\_timer.c, [606](#)
  - niietcm4\_timer.h, [612](#)
- UART, [337](#)
- UART\_BaudRate
  - UART\_Init\_TypeDef, [395](#)
- UART\_BaudRateDivConfig
  - Функции, [207](#)
  - Приватные функции, [341](#)
  - niietcm4\_uart.c, [615](#)
  - niietcm4\_uart.h, [630](#)
- UART\_Break
  - Функции, [207](#)
  - Приватные функции, [341](#)
  - niietcm4\_uart.c, [616](#)
  - niietcm4\_uart.h, [631](#)
- UART\_CTSEn
  - UART\_ModemInit\_TypeDef, [397](#)
- UART\_ClkFreq
  - UART\_Init\_TypeDef, [396](#)
- UART\_Cmd
  - Функции, [208](#)
  - Приватные функции, [341](#)
  - niietcm4\_uart.c, [616](#)
  - niietcm4\_uart.h, [631](#)
- UART\_DMABlkOnErrCmd
  - Настройка DMA, [219](#)
  - Приватные функции, [343](#)
  - niietcm4\_uart.c, [617](#)
  - niietcm4\_uart.h, [632](#)
- UART\_DMACmd
  - Настройка DMA, [219](#)
  - Приватные функции, [343](#)
  - niietcm4\_uart.c, [617](#)
  - niietcm4\_uart.h, [632](#)
- UART\_DataWidth
  - UART\_Init\_TypeDef, [396](#)
- UART\_DataWidth\_5
  - Типы, [199](#)
  - niietcm4\_uart.h, [629](#)
- UART\_DataWidth\_6
  - Типы, [199](#)
  - niietcm4\_uart.h, [629](#)
- UART\_DataWidth\_7
  - Типы, [199](#)
  - niietcm4\_uart.h, [629](#)
- UART\_DataWidth\_8
  - Типы, [199](#)
  - niietcm4\_uart.h, [629](#)
- UART\_DataWidth\_TypeDef
  - Типы, [199](#)
  - niietcm4\_uart.h, [629](#)
- UART\_DeInit
  - Инициализация и деинициализация, [209](#)
  - Приватные функции, [341](#)
  - niietcm4\_uart.c, [616](#)
  - niietcm4\_uart.h, [631](#)
- UART\_Dir\_Rx
  - Типы, [199](#)
  - niietcm4\_uart.h, [630](#)
- UART\_Dir\_Tx
  - Типы, [199](#)
  - niietcm4\_uart.h, [630](#)
- UART\_Dir\_Typedef
  - Типы, [199](#)
  - niietcm4\_uart.h, [629](#)
- UART\_Error\_All
  - Ошибки приемника UART, [206](#)
  - niietcm4\_uart.h, [626](#)
- UART\_Error\_Break
  - Ошибки приемника UART, [206](#)
  - niietcm4\_uart.h, [626](#)
- UART\_Error\_Frame
  - Ошибки приемника UART, [206](#)

- niietcm4\_uart.h, 627
- UART\_Error\_Overflow
  - Ошибки приемника UART, 206
  - niietcm4\_uart.h, 627
- UART\_Error\_Parity
  - Ошибки приемника UART, 206
  - niietcm4\_uart.h, 627
- UART\_ErrorStatus
  - Прием и передача, 212
  - Приватные функции, 343
  - niietcm4\_uart.c, 617
  - niietcm4\_uart.h, 632
- UART\_ErrorStatusClear
  - Прием и передача, 212
  - Приватные функции, 344
  - niietcm4\_uart.c, 618
  - niietcm4\_uart.h, 633
- UART\_FIFOEn
  - UART\_Init\_TypeDef, 396
- UART\_FIFOLevel\_1\_2
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_FIFOLevel\_1\_4
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_FIFOLevel\_1\_8
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_FIFOLevel\_3\_4
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_FIFOLevel\_7\_8
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_FIFOLevel\_TypeDef
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_FIFOLevelRx
  - UART\_Init\_TypeDef, 396
- UART\_FIFOLevelTx
  - UART\_Init\_TypeDef, 396
- UART\_Flag\_All
  - Флаги работы UART, 204
  - niietcm4\_uart.h, 627
- UART\_Flag\_Busy
  - Флаги работы UART, 204
  - niietcm4\_uart.h, 627
- UART\_Flag\_InvCTS
  - Флаги работы UART, 204
  - niietcm4\_uart.h, 627
- UART\_Flag\_InvDCD
  - Флаги работы UART, 204
  - niietcm4\_uart.h, 627
- UART\_Flag\_InvDSR
  - Флаги работы UART, 204
  - niietcm4\_uart.h, 627
- UART\_Flag\_InvRI
  - Флаги работы UART, 204
- niietcm4\_uart.h, 627
- UART\_Flag\_RxFIFOEmpty
  - Флаги работы UART, 204
  - niietcm4\_uart.h, 628
- UART\_Flag\_RxFIFOFull
  - Флаги работы UART, 205
  - niietcm4\_uart.h, 628
- UART\_Flag\_TxFIFOEmpty
  - Флаги работы UART, 205
  - niietcm4\_uart.h, 628
- UART\_Flag\_TxFIFOFull
  - Флаги работы UART, 205
  - niietcm4\_uart.h, 628
- UART\_FlagStatus
  - Прием и передача, 212
  - Приватные функции, 344
  - niietcm4\_uart.c, 618
  - niietcm4\_uart.h, 633
- UART\_ITCmd
  - Прерывания, 216
  - Приватные функции, 345
  - niietcm4\_uart.c, 619
  - niietcm4\_uart.h, 634
- UART\_ITFIFOLevelConfig
  - Прерывания, 216
  - Приватные функции, 345
  - niietcm4\_uart.c, 619
  - niietcm4\_uart.h, 634
- UART\_ITMaskedStatus
  - Прерывания, 217
  - Приватные функции, 345
  - niietcm4\_uart.c, 619
  - niietcm4\_uart.h, 634
- UART\_ITRawStatus
  - Прерывания, 217
  - Приватные функции, 347
  - niietcm4\_uart.c, 620
  - niietcm4\_uart.h, 635
- UART\_ITSource\_All
  - Источники прерываний UART, 202
  - niietcm4\_uart.h, 628
- UART\_ITSource\_ChangeCTS
  - Источники прерываний UART, 202
  - niietcm4\_uart.h, 628
- UART\_ITSource\_ChangeDCD
  - Источники прерываний UART, 202
  - niietcm4\_uart.h, 628
- UART\_ITSource\_ChangeDSR
  - Источники прерываний UART, 202
  - niietcm4\_uart.h, 628
- UART\_ITSource\_ChangeRI
  - Источники прерываний UART, 202
  - niietcm4\_uart.h, 628
- UART\_ITSource\_ErrorBreak
  - Источники прерываний UART, 202
  - niietcm4\_uart.h, 628
- UART\_ITSource\_ErrorFrame
  - Источники прерываний UART, 203

- niietcm4\_uart.h, 629
- UART\_ITSource\_ErrorOverflow
  - Источники прерываний UART, 203
  - niietcm4\_uart.h, 629
- UART\_ITSource\_ErrorParity
  - Источники прерываний UART, 203
  - niietcm4\_uart.h, 629
- UART\_ITSource\_RecieveTimeout
  - Источники прерываний UART, 203
  - niietcm4\_uart.h, 629
- UART\_ITSource\_RxFIFOLevel
  - Источники прерываний UART, 203
  - niietcm4\_uart.h, 629
- UART\_ITSource\_TxFIFOLevel
  - Источники прерываний UART, 203
  - niietcm4\_uart.h, 629
- UART\_ITStatusClear
  - Прерывания, 217
  - Приватные функции, 347
  - niietcm4\_uart.c, 620
  - niietcm4\_uart.h, 635
- UART\_Init
  - Инициализация и деинициализация, 209
  - Приватные функции, 344
  - niietcm4\_uart.c, 618
  - niietcm4\_uart.h, 633
- UART\_Init\_TypeDef, 395
  - UART\_BaudRate, 395
  - UART\_ClkFreq, 396
  - UART\_DataWidth, 396
  - UART\_FIFOEn, 396
  - UART\_FIFOLevelRx, 396
  - UART\_FIFOLevelTx, 396
  - UART\_ParityBit, 396
  - UART\_RxEn, 396
  - UART\_StopBit, 397
  - UART\_TxEn, 397
- UART\_InvDTR
  - UART\_ModemInit\_TypeDef, 397
- UART\_InvRTS
  - UART\_ModemInit\_TypeDef, 398
- UART\_ModemConfig
  - Приватные функции, 347
  - Режим модема, 215
  - niietcm4\_uart.c, 620
  - niietcm4\_uart.h, 635
- UART\_ModemInit\_TypeDef, 397
  - UART\_CTSEn, 397
  - UART\_InvDTR, 397
  - UART\_InvRTS, 398
  - UART\_RTSEn, 398
- UART\_ModemStructInit
  - Приватные функции, 348
  - Режим модема, 215
  - niietcm4\_uart.c, 621
  - niietcm4\_uart.h, 636
- UART\_ParityBit
  - UART\_Init\_TypeDef, 396
- UART\_ParityBit\_Disable
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_ParityBit\_Even
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_ParityBit\_High
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_ParityBit\_Low
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_ParityBit\_Odd
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_ParityBit\_TypeDef
  - Типы, 199
  - niietcm4\_uart.h, 630
- UART\_RTSEn
  - UART\_ModemInit\_TypeDef, 398
- UART\_RecieveData
  - Прием и передача, 214
  - Приватные функции, 348
  - niietcm4\_uart.c, 621
  - niietcm4\_uart.h, 636
- UART\_RxEn
  - UART\_Init\_TypeDef, 396
- UART\_SendData
  - Прием и передача, 214
  - Приватные функции, 348
  - niietcm4\_uart.c, 621
  - niietcm4\_uart.h, 636
- UART\_StopBit
  - UART\_Init\_TypeDef, 397
- UART\_StopBit\_1
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_StopBit\_2
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_StopBit\_TypeDef
  - Типы, 200
  - niietcm4\_uart.h, 630
- UART\_StructInit
  - Инициализация и деинициализация, 209
  - Приватные функции, 349
  - niietcm4\_uart.c, 622
  - niietcm4\_uart.h, 637
- UART\_TxEn
  - UART\_Init\_TypeDef, 397
- USERFLASH, 350
  - USERFLASH\_FullErase
    - Основная область флеш, 226
    - Приватные функции, 353
    - niietcm4\_userflash.c, 638
    - niietcm4\_userflash.h, 644
  - USERFLASH\_INFO\_PAGE\_SIZE\_BYTES
    - Информационная область флеш, 222

- [niietcm4\\_userflash.h](#), [643](#)
- USERFLASH\_INFO\_PAGE\_TOTAL
  - Информационная область флеш, [222](#)
- [niietcm4\\_userflash.h](#), [643](#)
- USERFLASH\_INFO\_TOTAL\_BYTES
  - Информационная область флеш, [222](#)
- [niietcm4\\_userflash.h](#), [643](#)
- USERFLASH\_ITCmd
  - Функции, [224](#)
  - Приватные функции, [354](#)
- [niietcm4\\_userflash.c](#), [639](#)
  - [niietcm4\\_userflash.h](#), [645](#)
- USERFLASH\_Info\_PageErase
  - Информационная область флеш, [228](#)
  - Приватные функции, [353](#)
- [niietcm4\\_userflash.c](#), [638](#)
  - [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_Info\_Read
  - Информационная область флеш, [228](#)
  - Приватные функции, [354](#)
- [niietcm4\\_userflash.c](#), [639](#)
  - [niietcm4\\_userflash.h](#), [645](#)
- USERFLASH\_Info\_Write
  - Информационная область флеш, [228](#)
  - Приватные функции, [354](#)
- [niietcm4\\_userflash.c](#), [639](#)
  - [niietcm4\\_userflash.h](#), [645](#)
- USERFLASH\_Init
  - Функции, [224](#)
  - Приватные функции, [354](#)
- [niietcm4\\_userflash.c](#), [639](#)
  - [niietcm4\\_userflash.h](#), [645](#)
- USERFLASH\_OperationStatus
  - Функции, [224](#)
  - Приватные функции, [355](#)
- [niietcm4\\_userflash.c](#), [640](#)
  - [niietcm4\\_userflash.h](#), [646](#)
- USERFLASH\_OperationStatusClear
  - Функции, [225](#)
  - Приватные функции, [355](#)
- [niietcm4\\_userflash.c](#), [640](#)
  - [niietcm4\\_userflash.h](#), [646](#)
- USERFLASH\_PAGE\_SIZE\_BYTES
  - Основная область флеш, [221](#)
- [niietcm4\\_userflash.h](#), [643](#)
- USERFLASH\_PAGE\_TOTAL
  - Основная область флеш, [221](#)
- [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_PageErase
  - Основная область флеш, [226](#)
  - Приватные функции, [355](#)
- [niietcm4\\_userflash.c](#), [640](#)
  - [niietcm4\\_userflash.h](#), [646](#)
- USERFLASH\_Read
  - Основная область флеш, [226](#)
  - Приватные функции, [355](#)
- [niietcm4\\_userflash.c](#), [640](#)
  - [niietcm4\\_userflash.h](#), [646](#)
- USERFLASH\_Status\_Complete
  - Типы, [223](#)
- [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_Status\_Error
  - Типы, [223](#)
- [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_Status\_None
  - Типы, [223](#)
- [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_Status\_TypeDef
  - Типы, [223](#)
- [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_TOTAL\_BYTES
  - Основная область флеш, [221](#)
- [niietcm4\\_userflash.h](#), [644](#)
- USERFLASH\_Write
  - Основная область флеш, [227](#)
  - Приватные функции, [356](#)
- [niietcm4\\_userflash.c](#), [641](#)
  - [niietcm4\\_userflash.h](#), [647](#)
- WATCHDOG, [357](#)
- WATCHDOG\_Cmd
  - Конфигурация, [233](#)
  - Приватные функции, [360](#)
- [niietcm4\\_watchdog.c](#), [649](#)
  - [niietcm4\\_watchdog.h](#), [652](#)
- WATCHDOG\_GetCounter
  - Конфигурация, [233](#)
  - Приватные функции, [360](#)
- [niietcm4\\_watchdog.c](#), [649](#)
  - [niietcm4\\_watchdog.h](#), [652](#)
- WATCHDOG\_GetReload
  - Конфигурация, [233](#)
  - Приватные функции, [360](#)
- [niietcm4\\_watchdog.c](#), [649](#)
  - [niietcm4\\_watchdog.h](#), [652](#)
- WATCHDOG\_ITMaskedStatus
  - Прерывания, [235](#)
  - Приватные функции, [361](#)
- [niietcm4\\_watchdog.c](#), [649](#)
  - [niietcm4\\_watchdog.h](#), [652](#)
- WATCHDOG\_ITRawStatus
  - Прерывания, [235](#)
  - Приватные функции, [361](#)
- [niietcm4\\_watchdog.c](#), [649](#)
  - [niietcm4\\_watchdog.h](#), [653](#)
- WATCHDOG\_ITStatusClear
  - Прерывания, [235](#)
  - Приватные функции, [361](#)
- [niietcm4\\_watchdog.c](#), [649](#)
  - [niietcm4\\_watchdog.h](#), [653](#)
- WATCHDOG\_Lock\_Value
  - Приватные константы, [359](#)
- [niietcm4\\_watchdog.c](#), [648](#)
- WATCHDOG\_LockCmd
  - Конфигурация, [234](#)
  - Приватные функции, [361](#)
- [niietcm4\\_watchdog.c](#), [650](#)

- [niietcm4\\_watchdog.h, 653](#)
- WATCHDOG\_RstCmd
  - Конфигурация, [234](#)
  - Приватные функции, [361](#)
  - [niietcm4\\_watchdog.c, 650](#)
  - [niietcm4\\_watchdog.h, 653](#)
- WATCHDOG\_SetReload
  - Конфигурация, [234](#)
  - Приватные функции, [362](#)
  - [niietcm4\\_watchdog.c, 650](#)
  - [niietcm4\\_watchdog.h, 653](#)
- WATCHDOG\_Unlock\_Value
  - Приватные константы, [359](#)
  - [niietcm4\\_watchdog.c, 648](#)