

НИЕТСМ4 PD
v0.7.0

Создано системой Doxygen 1.8.10

Чт 24 Дек 2015 08:52:37

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	17
6.2.2.3	ADC_Channel_10	17
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	18
6.2.2.13	ADC_Channel_2	18
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	19
6.2.2.22	ADC_Channel_7	19
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	21
6.3.2.3	ADC_DC_10	21
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	22
6.3.2.13	ADC_DC_2	22
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	23
6.3.2.22	ADC_DC_7	23
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	25
6.4.2.6	ADC_SEQ_5	25
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	29
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	31
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	32
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	34
6.5.3.6	ADC_Measure_TypeDef	34
6.5.3.7	ADC_Mode_TypeDef	35
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	38
6.6.2	Функции	38
6.6.2.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	38
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)	39
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)	40
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)	41
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)	42
6.6.2.14	ADC_SEQ_SWReq()	42
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)	45
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)	47
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	50
6.11.1	Подробное описание	50
6.11.2	Функции	50
6.11.2.1	ADC_SEQ_DMACmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	50
6.11.2.2	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)	50
6.11.2.3	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	51
6.11.2.4	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	51
6.12	Конфигурация прерываний	52
6.12.1	Подробное описание	52
6.13	Цифровые компараторы	53
6.13.1	Подробное описание	53
6.13.2	Функции	53
6.13.2.1	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	53
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)	54
6.13.2.3	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	54
6.13.2.4	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	55
6.13.2.5	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	56
6.13.2.6	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	56
6.13.2.7	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_Module)	56
6.14	Секвенсоры	58
6.14.1	Подробное описание	58
6.14.2	Функции	58

6.14.2.1	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module)	58
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)	59
6.14.2.4	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_↵ _Module)	59
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)	60
6.14.2.7	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SE↵ Q_Module)	60
6.15	Константы	61
6.15.1	Подробное описание	61
6.16	Основная область флеш	62
6.16.1	Подробное описание	62
6.16.2	Макросы	62
6.16.2.1	BOOTFLASH_PAGE_SIZE_BYTES	62
6.16.2.2	BOOTFLASH_PAGE_TOTAL	62
6.16.2.3	BOOTFLASH_TOTAL_BYTES	62
6.17	Информационная область флеш	63
6.17.1	Подробное описание	63
6.17.2	Макросы	63
6.17.2.1	BOOTFLASH_INFO_PAGE_SIZE_BYTES	63
6.17.2.2	BOOTFLASH_INFO_PAGE_TOTAL	63
6.17.2.3	BOOTFLASH_INFO_TOTAL_BYTES	63
6.18	Типы	64
6.18.1	Подробное описание	64
6.18.2	Макросы	64
6.18.2.1	IS_BOOTFLASH_STATUS	64
6.18.3	Перечисления	64
6.18.3.1	BOOTFLASH_Status_TypeDef	64
6.19	Функции	65
6.19.1	Подробное описание	65
6.19.2	Функции	65
6.19.2.1	BOOTFLASH_Init(uint32_t SysClkFreq)	65
6.19.2.2	BOOTFLASH_ITCmd(FunctionalState State)	65
6.19.2.3	BOOTFLASH_OperationStatus()	66
6.19.2.4	BOOTFLASH_OperationStatusClear()	66

6.20	Основная область флеш	67
6.20.1	Подробное описание	67
6.20.2	Функции	67
6.20.2.1	BOOTFLASH_FullErase()	67
6.20.2.2	BOOTFLASH_PageErase(uint32_t PageNum)	67
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	Константы	72
6.22.1	Подробное описание	72
6.23	Маски для CHANNEL_CFG	73
6.23.1	Подробное описание	73
6.23.2	Макросы	73
6.23.2.1	CHANNEL_CFG_CYCLE_CTRL_Msk	73
6.23.2.2	CHANNEL_CFG_CYCLE_CTRL_Pos	73
6.23.2.3	CHANNEL_CFG_DST_INC_Msk	74
6.23.2.4	CHANNEL_CFG_DST_INC_Pos	74
6.23.2.5	CHANNEL_CFG_DST_PROT_CTRL_Msk	74
6.23.2.6	CHANNEL_CFG_DST_PROT_CTRL_Pos	74
6.23.2.7	CHANNEL_CFG_DST_SIZE_Msk	74
6.23.2.8	CHANNEL_CFG_DST_SIZE_Pos	74
6.23.2.9	CHANNEL_CFG_N_MINUS_1_Msk	74
6.23.2.10	CHANNEL_CFG_N_MINUS_1_Pos	74
6.23.2.11	CHANNEL_CFG_NEXT_USEBURST_Msk	74
6.23.2.12	CHANNEL_CFG_NEXT_USEBURST_Pos	75
6.23.2.13	CHANNEL_CFG_R_POWER_Msk	75
6.23.2.14	CHANNEL_CFG_R_POWER_Pos	75
6.23.2.15	CHANNEL_CFG_SRC_INC_Msk	75
6.23.2.16	CHANNEL_CFG_SRC_INC_Pos	75
6.23.2.17	CHANNEL_CFG_SRC_PROT_CTRL_Msk	75
6.23.2.18	CHANNEL_CFG_SRC_PROT_CTRL_Pos	75
6.23.2.19	CHANNEL_CFG_SRC_SIZE_Msk	75
6.23.2.20	CHANNEL_CFG_SRC_SIZE_Pos	75
6.24	Маски каналов DMA	76
6.24.1	Подробное описание	76

6.24.2	Макросы	76
6.24.2.1	DMA_Channel_All	76
6.24.2.2	IS_GET_DMA_CHANNEL	76
6.25	Маски каналов по имени	78
6.25.1	Подробное описание	78
6.25.2	Макросы	78
6.25.2.1	DMA_Channel_ADCSEQ0	78
6.25.2.2	DMA_Channel_ADCSEQ1	79
6.25.2.3	DMA_Channel_ADCSEQ2	79
6.25.2.4	DMA_Channel_ADCSEQ3	79
6.25.2.5	DMA_Channel_ADCSEQ4	79
6.25.2.6	DMA_Channel_ADCSEQ5	79
6.25.2.7	DMA_Channel_ADCSEQ6	79
6.25.2.8	DMA_Channel_ADCSEQ7	79
6.25.2.9	DMA_Channel_SPI0_RX	79
6.25.2.10	DMA_Channel_SPI0_TX	79
6.25.2.11	DMA_Channel_SPI1_RX	79
6.25.2.12	DMA_Channel_SPI1_TX	80
6.25.2.13	DMA_Channel_SPI2_RX	80
6.25.2.14	DMA_Channel_SPI2_TX	80
6.25.2.15	DMA_Channel_SPI3_RX	80
6.25.2.16	DMA_Channel_SPI3_TX	80
6.25.2.17	DMA_Channel_UART0_RX	80
6.25.2.18	DMA_Channel_UART0_TX	80
6.25.2.19	DMA_Channel_UART1_RX	80
6.25.2.20	DMA_Channel_UART1_TX	80
6.25.2.21	DMA_Channel_UART2_RX	81
6.25.2.22	DMA_Channel_UART2_TX	81
6.25.2.23	DMA_Channel_UART3_RX	81
6.25.2.24	DMA_Channel_UART3_TX	81
6.26	Маски каналов по номеру	82
6.26.1	Подробное описание	82
6.26.2	Макросы	82
6.26.2.1	DMA_Channel_0	82
6.26.2.2	DMA_Channel_1	83
6.26.2.3	DMA_Channel_10	83
6.26.2.4	DMA_Channel_11	83
6.26.2.5	DMA_Channel_12	83
6.26.2.6	DMA_Channel_13	83
6.26.2.7	DMA_Channel_14	83

6.26.2.8 DMA_Channel_15	83
6.26.2.9 DMA_Channel_16	83
6.26.2.10 DMA_Channel_17	83
6.26.2.11 DMA_Channel_18	83
6.26.2.12 DMA_Channel_19	84
6.26.2.13 DMA_Channel_2	84
6.26.2.14 DMA_Channel_20	84
6.26.2.15 DMA_Channel_21	84
6.26.2.16 DMA_Channel_22	84
6.26.2.17 DMA_Channel_23	84
6.26.2.18 DMA_Channel_3	84
6.26.2.19 DMA_Channel_4	84
6.26.2.20 DMA_Channel_5	84
6.26.2.21 DMA_Channel_6	85
6.26.2.22 DMA_Channel_7	85
6.26.2.23 DMA_Channel_8	85
6.26.2.24 DMA_Channel_9	85
6.27 Типы	86
6.27.1 Подробное описание	87
6.27.2 Макросы	87
6.27.2.1 IS_DMA_ARBITRATION_RATE	87
6.27.2.2 IS_DMA_DATA_INC	88
6.27.2.3 IS_DMA_DATA_SIZE	88
6.27.2.4 IS_DMA_MODE	88
6.27.2.5 IS_DMA_STATE	88
6.27.3 Перечисления	89
6.27.3.1 DMA_ArbitrationRate_TypeDef	89
6.27.3.2 DMA_DataInc_TypeDef	89
6.27.3.3 DMA_DataSize_TypeDef	89
6.27.3.4 DMA_Mode_TypeDef	90
6.27.3.5 DMA_State_TypeDef	90
6.28 Функции	91
6.28.1 Подробное описание	91
6.29 Инициализация каналов DMA	92
6.29.1 Подробное описание	92
6.29.2 Функции	92
6.29.2.1 DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	92
6.29.2.2 DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA↵ _ChannelInit_TypeDef *DMA_ChannelInitStruct)	92

6.29.2.3	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	93
6.30	Инициализация контроллера DMA	94
6.30.1	Подробное описание	94
6.30.2	Функции	94
6.30.2.1	DMA_DeInit()	94
6.30.2.2	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	94
6.30.2.3	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	95
6.31	Конфигурация контроллера DMA	96
6.31.1	Подробное описание	96
6.31.2	Функции	96
6.31.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	96
6.31.2.2	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	97
6.31.2.3	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	97
6.31.2.4	DMA_MasterEnableCmd(FunctionalState State)	97
6.31.2.5	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	98
6.31.2.6	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	98
6.31.2.7	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	98
6.31.2.8	DMA_SWRequestCmd(uint32_t DMA_Channel)	99
6.31.2.9	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	99
6.32	Статусная информация	100
6.32.1	Подробное описание	100
6.32.2	Функции	100
6.32.2.1	DMA_ClearErrorStatus()	100
6.32.2.2	DMA_ErrorStatus()	100
6.32.2.3	DMA_MasterEnableStatus()	101
6.32.2.4	DMA_StateStatus()	102
6.32.2.5	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	102
6.33	Константы	103
6.33.1	Подробное описание	103
6.34	Маски адреса	104
6.34.1	Подробное описание	104
6.34.2	Макросы	104
6.34.2.1	EXTMEM_CEMask_Addr_11	104
6.34.2.2	EXTMEM_CEMask_Addr_11_19	104
6.34.2.3	EXTMEM_CEMask_Addr_12	104
6.34.2.4	EXTMEM_CEMask_Addr_13	105
6.34.2.5	EXTMEM_CEMask_Addr_14	105
6.34.2.6	EXTMEM_CEMask_Addr_15	105

6.34.2.7	EXTMEM_CEMask_Addr_16	105
6.34.2.8	EXTMEM_CEMask_Addr_17	105
6.34.2.9	EXTMEM_CEMask_Addr_18	105
6.34.2.10	EXTMEM_CEMask_Addr_19	105
6.35	Типы	106
6.35.1	Подробное описание	107
6.35.2	Макросы	107
6.35.2.1	IS_EXTMEM_READ_WAITSTATE	107
6.35.2.2	IS_EXTMEM_RW_WAITSTATE	107
6.35.2.3	IS_EXTMEM_WIDTH	107
6.35.2.4	IS_EXTMEM_WRITE_WAITSTATE	107
6.35.3	Перечисления	108
6.35.3.1	EXTMEM_ReadWaitState_TypeDef	108
6.35.3.2	EXTMEM_RWWaitState_TypeDef	108
6.35.3.3	EXTMEM_Width_TypeDef	109
6.35.3.4	EXTMEM_WriteWaitState_TypeDef	109
6.36	Функции	110
6.36.1	Подробное описание	110
6.36.2	Функции	110
6.36.2.1	EXTMEM_DeInit()	110
6.36.2.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	110
6.36.2.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	111
6.37	Типы	113
6.37.1	Подробное описание	114
6.37.2	Макросы	114
6.37.2.1	IS_GPIO_ALT_FUNC	114
6.37.2.2	IS_GPIO_DIR	114
6.37.2.3	IS_GPIO_INT_POL	115
6.37.2.4	IS_GPIO_INT_TYPE	115
6.37.2.5	IS_GPIO_LOAD	115
6.37.2.6	IS_GPIO_MODE	115
6.37.2.7	IS_GPIO_OUT	115
6.37.2.8	IS_GPIO_OUT_MODE	116
6.37.2.9	IS_GPIO_PULLUP	116
6.37.2.10	IS_GPIO_QUAL	116
6.37.2.11	IS_GPIO_QUAL_MODE	116
6.37.2.12	IS_GPIO_SYNC	116
6.37.3	Перечисления	117
6.37.3.1	BitAction	117
6.37.3.2	GPIO_AltFunc_TypeDef	117

6.37.3.3	GPIO_Dir_TypeDef	117
6.37.3.4	GPIO_IntPol_TypeDef	117
6.37.3.5	GPIO_IntType_TypeDef	117
6.37.3.6	GPIO_Load_TypeDef	118
6.37.3.7	GPIO_Mode_TypeDef	118
6.37.3.8	GPIO_Out_TypeDef	118
6.37.3.9	GPIO_OutMode_TypeDef	118
6.37.3.10	GPIO_PullUp_TypeDef	118
6.37.3.11	GPIO_Qual_TypeDef	119
6.37.3.12	GPIO_QualMode_TypeDef	119
6.37.3.13	GPIO_Sync_TypeDef	119
6.38	Константы	120
6.38.1	Подробное описание	120
6.39	Маски пинов	121
6.39.1	Подробное описание	121
6.39.2	Макросы	121
6.39.2.1	GPIO_Pin_0	121
6.39.2.2	GPIO_Pin_0_3	122
6.39.2.3	GPIO_Pin_0_7	122
6.39.2.4	GPIO_Pin_1	122
6.39.2.5	GPIO_Pin_10	122
6.39.2.6	GPIO_Pin_11	122
6.39.2.7	GPIO_Pin_12	122
6.39.2.8	GPIO_Pin_12_15	122
6.39.2.9	GPIO_Pin_13	122
6.39.2.10	GPIO_Pin_14	122
6.39.2.11	GPIO_Pin_15	122
6.39.2.12	GPIO_Pin_2	123
6.39.2.13	GPIO_Pin_3	123
6.39.2.14	GPIO_Pin_4	123
6.39.2.15	GPIO_Pin_4_7	123
6.39.2.16	GPIO_Pin_5	123
6.39.2.17	GPIO_Pin_6	123
6.39.2.18	GPIO_Pin_7	123
6.39.2.19	GPIO_Pin_8	123
6.39.2.20	GPIO_Pin_8_11	123
6.39.2.21	GPIO_Pin_8_15	124
6.39.2.22	GPIO_Pin_9	124
6.39.2.23	GPIO_Pin_All	124
6.39.2.24	IS_GET_GPIO_PIN	124

6.40	Функции	125
6.40.1	Подробное описание	125
6.41	Инициализация и деинициализация	126
6.41.1	Подробное описание	126
6.41.2	Функции	126
6.41.2.1	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	126
6.41.2.2	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	126
6.41.2.3	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	127
6.42	Чтение и запись	128
6.42.1	Подробное описание	128
6.43	Чтение	129
6.43.1	Подробное описание	129
6.43.2	Функции	129
6.43.2.1	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	129
6.43.2.2	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	129
6.43.2.3	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	130
6.44	Запись	131
6.44.1	Подробное описание	131
6.44.2	Функции	131
6.44.2.1	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	131
6.44.2.2	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	131
6.44.2.3	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	132
6.45	Битовые операции	133
6.45.1	Подробное описание	133
6.45.2	Функции	133
6.45.2.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	133
6.45.2.2	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	133
6.45.2.3	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	134
6.46	Фильтрация	135
6.46.1	Подробное описание	135
6.46.2	Функции	135
6.46.2.1	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	135
6.46.2.2	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	135
6.46.2.3	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	136
6.47	Прерывания	137
6.47.1	Подробное описание	137

6.47.2	Функции	137
6.47.2.1	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	137
6.47.2.2	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	137
6.47.2.3	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	138
6.48	Константы	139
6.48.1	Подробное описание	139
6.48.2	Макросы	139
6.48.2.1	RCC_CLK_CHANGE_TIMEOUT	139
6.49	Типы	140
6.49.1	Подробное описание	142
6.49.2	Макросы	142
6.49.2.1	IS_RCC_ADC_CLK	142
6.49.2.2	IS_RCC_PERIPH_CLK	142
6.49.2.3	IS_RCC_PLL_NO	143
6.49.2.4	IS_RCC_PLL_REF	143
6.49.2.5	IS_RCC_SPI_CLK	143
6.49.2.6	IS_RCC_SYS_CLK	143
6.49.2.7	IS_RCC_UART_CLK	144
6.49.2.8	IS_RCC_USB_CLK	144
6.49.2.9	IS_RCC_USB_FREQ	144
6.49.3	Перечисления	144
6.49.3.1	RCC_ADCClk_TypeDef	144
6.49.3.2	RCC_PeriphClk_TypeDef	145
6.49.3.3	RCC_PeriphRst_TypeDef	145
6.49.3.4	RCC_PLLNO_TypeDef	146
6.49.3.5	RCC_PLLRef_TypeDef	146
6.49.3.6	RCC_SPIClk_TypeDef	147
6.49.3.7	RCC_SysClk_TypeDef	147
6.49.3.8	RCC_UARTClk_TypeDef	147
6.49.3.9	RCC_USBClk_TypeDef	147
6.49.3.10	RCC_USBFreq_TypeDef	148
6.50	Функции	149
6.50.1	Подробное описание	149
6.50.2	Функции	149
6.50.2.1	RCC_SysClkDiv2Out(FunctionalState State)	149
6.51	Конфигурация PLL	151
6.51.1	Подробное описание	151
6.51.2	Функции	151

6.51.2.1	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	151
6.51.2.2	RCC_PLLDeInit()	152
6.51.2.3	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	152
6.51.2.4	RCC_PLLPowerDownCmd(FunctionalState State)	153
6.51.2.5	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	153
6.52	Управление тактированием	154
6.52.1	Подробное описание	154
6.52.2	Функции	154
6.52.2.1	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	154
6.52.2.2	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	155
6.52.2.3	RCC_SysClkStatus()	155
6.53	Тактирование USB	156
6.53.1	Подробное описание	156
6.53.2	Функции	156
6.53.2.1	RCC_USBClkCmd(FunctionalState State)	156
6.53.2.2	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	156
6.54	Тактирование UART	158
6.54.1	Подробное описание	158
6.54.2	Функции	158
6.54.2.1	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	158
6.54.2.2	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	158
6.54.2.3	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	159
6.55	Тактирование SPI	160
6.55.1	Подробное описание	160
6.55.2	Функции	160
6.55.2.1	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	160
6.55.2.2	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	160
6.55.2.3	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	161
6.56	Тактирование ADC	162
6.56.1	Подробное описание	162
6.56.2	Функции	162
6.56.2.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	162
6.56.2.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	162

6.57	Управление сбросом	164
6.57.1	Подробное описание	164
6.57.2	Функции	164
6.57.2.1	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	164
6.58	Типы	165
6.58.1	Подробное описание	166
6.58.2	Макросы	166
6.58.2.1	IS_RTC_FORMAT	166
6.58.2.2	IS_RTC_MONTH	166
6.58.2.3	IS_RTC_WEEKDAY	166
6.58.3	Перечисления	167
6.58.3.1	RTC_Format_TypeDef	167
6.58.3.2	RTC_Month_TypeDef	167
6.58.3.3	RTC_Weekday_TypeDef	167
6.59	Функции	168
6.59.1	Подробное описание	168
6.60	Типы	169
6.60.1	Подробное описание	169
6.60.2	Макросы	169
6.60.2.1	IS_TIMER_EXT_INPUT	169
6.60.3	Перечисления	169
6.60.3.1	TIMER_ExtInput_TypeDef	169
6.61	Константы	171
6.62	Функции	172
6.62.1	Подробное описание	172
6.63	Конфигурация	173
6.63.1	Подробное описание	173
6.63.2	Функции	173
6.63.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	173
6.63.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	174
6.63.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	174
6.63.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	174
6.63.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	175
6.63.2.6	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	175
6.63.2.7	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	175
6.63.2.8	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	176

6.64	Прерывания	177
6.64.1	Подробное описание	177
6.64.2	Функции	177
6.64.2.1	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	177
6.64.2.2	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	177
6.64.2.3	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	178
6.65	Типы	179
6.65.1	Подробное описание	180
6.65.2	Макросы	180
6.65.2.1	IS_UART_DATA_WIDTH	180
6.65.2.2	IS_UART_DIR	181
6.65.2.3	IS_UART_ERROR	181
6.65.2.4	IS_UART_FIFO_LEVEL	181
6.65.2.5	IS_UART_FLAG	181
6.65.2.6	IS_UART_GET_IT_SOURCE	182
6.65.2.7	IS_UART_PARITY_BIT	182
6.65.2.8	IS_UART_STOP_BIT	182
6.65.3	Перечисления	182
6.65.3.1	UART_DataWidth_TypeDef	182
6.65.3.2	UART_Dir_Typedef	183
6.65.3.3	UART_Error_Typedef	183
6.65.3.4	UART_FIFOLevel_TypeDef	183
6.65.3.5	UART_Flag_Typedef	184
6.65.3.6	UART_ITSource_Typedef	184
6.65.3.7	UART_ParityBit_TypeDef	184
6.65.3.8	UART_StopBit_TypeDef	185
6.66	Константы	186
6.67	Функции	187
6.67.1	Подробное описание	187
6.67.2	Функции	187
6.67.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	187
6.67.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	188
6.67.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	188
6.68	Инициализация и деинициализация	189
6.68.1	Подробное описание	189
6.68.2	Функции	189
6.68.2.1	UART_DeInit(NT_UART_TypeDef *UARTx)	189
6.68.2.2	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	189

6.68.2.3	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	190
6.69	Прием и передача	191
6.69.1	Подробное описание	191
6.69.2	Функции	191
6.69.2.1	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error)	191
6.69.2.2	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	191
6.69.2.3	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag)	192
6.69.2.4	UART_RecieveData(NT_UART_TypeDef *UARTx)	192
6.69.2.5	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	192
6.70	Режим модема	193
6.70.1	Подробное описание	193
6.70.2	Функции	193
6.70.2.1	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	193
6.70.2.2	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	193
6.71	Прерывания	195
6.71.1	Подробное описание	195
6.71.2	Функции	195
6.71.2.1	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)	195
6.71.2.2	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_ITDir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	196
6.71.2.3	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	196
6.71.2.4	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	196
6.71.2.5	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	197
6.72	Настройка DMA	198
6.72.1	Подробное описание	198
6.72.2	Функции	198
6.72.2.1	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	198
6.72.2.2	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	198
6.73	Константы	200
6.73.1	Подробное описание	200
6.74	Основная область флеш	201
6.74.1	Подробное описание	201
6.74.2	Макросы	201

6.74.2.1	USERFLASH_PAGE_SIZE_BYTES	201
6.74.2.2	USERFLASH_PAGE_TOTAL	201
6.74.2.3	USERFLASH_TOTAL_BYTES	201
6.75	Информационная область флеш	202
6.75.1	Подробное описание	202
6.75.2	Макросы	202
6.75.2.1	USERFLASH_INFO_PAGE_SIZE_BYTES	202
6.75.2.2	USERFLASH_INFO_PAGE_TOTAL	202
6.75.2.3	USERFLASH_INFO_TOTAL_BYTES	202
6.76	Типы	203
6.76.1	Подробное описание	203
6.76.2	Макросы	203
6.76.2.1	IS_USERFLASH_STATUS	203
6.76.3	Перечисления	203
6.76.3.1	USERFLASH_Status_TypeDef	203
6.77	Функции	204
6.77.1	Подробное описание	204
6.77.2	Функции	204
6.77.2.1	USERFLASH_Init(uint32_t SysClkFreq)	204
6.77.2.2	USERFLASH_ITCmd(FunctionalState State)	204
6.77.2.3	USERFLASH_OperationStatus()	205
6.77.2.4	USERFLASH_OperationStatusClear()	205
6.78	Основная область флеш	206
6.78.1	Подробное описание	206
6.78.2	Функции	206
6.78.2.1	USERFLASH_FullErase()	206
6.78.2.2	USERFLASH_PageErase(uint32_t PageNum)	206
6.78.2.3	USERFLASH_Read(uint32_t Address)	207
6.78.2.4	USERFLASH_Write(uint32_t Address, uint32_t Data)	207
6.79	Информационная область флеш	208
6.79.1	Подробное описание	208
6.79.2	Функции	208
6.79.2.1	USERFLASH_Info_PageErase(uint32_t PageNum)	208
6.79.2.2	USERFLASH_Info_Read(uint32_t Address)	208
6.79.2.3	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	209
6.80	ADC	210
6.80.1	Подробное описание	210
6.81	Приватные данные	212
6.81.1	Подробное описание	212
6.82	Приватные константы	213

6.82.1	Подробное описание	213
6.83	Начальные значения регистров	214
6.84	Приватные функции	215
6.84.1	Подробное описание	217
6.84.2	Функции	217
6.84.2.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	217
6.84.2.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	217
6.84.2.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module)	217
6.84.2.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↵Module)	218
6.84.2.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↵DC_DC_Init_TypeDef *ADC_DC_InitStruct)	218
6.84.2.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	218
6.84.2.7	ADC_DC_ITConfig(ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition↵_TypeDef ADC_DC_Condition)	219
6.84.2.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↵Module, FunctionalState State)	219
6.84.2.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↵Module, FunctionalState State)	220
6.84.2.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC_↵Module)	221
6.84.2.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↵Module)	221
6.84.2.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↵Module)	221
6.84.2.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	222
6.84.2.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	222
6.84.2.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC_↵Module)	222
6.84.2.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	223
6.84.2.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↵Def *ADC_InitStruct)	223
6.84.2.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	223
6.84.2.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	224
6.84.2.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↵Module, FunctionalState State)	224
6.84.2.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_↵Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)	224
6.84.2.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↵SEQ_Module)	225

6.84.2.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	225
6.84.2.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	225
6.84.2.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	226
6.84.2.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	226
6.84.2.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	226
6.84.2.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	226
6.84.2.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	227
6.84.2.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	227
6.84.2.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	227
6.84.2.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	228
6.84.2.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	228
6.84.2.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	228
6.84.2.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	229
6.84.2.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	229
6.84.2.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	229
6.84.2.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	230
6.84.2.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	230
6.84.2.40	ADC_SEQ_SWReq()	230
6.84.2.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	231
6.85	BOOTFLASH	233
6.85.1	Подробное описание	233
6.86	Приватные данные	234
6.86.1	Подробное описание	234
6.87	Приватные константы	235
6.88	Приватные функции	236
6.88.1	Подробное описание	236
6.88.2	Функции	236

6.88.2.1	BOOTFLASH_FullErase()	236
6.88.2.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	237
6.88.2.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	238
6.88.2.4	BOOTFLASH_Init(uint32_t SysClkFreq)	238
6.88.2.5	BOOTFLASH_ITCmd(FunctionalState State)	238
6.88.2.6	BOOTFLASH_OperationStatus()	239
6.88.2.7	BOOTFLASH_OperationStatusClear()	239
6.88.2.8	BOOTFLASH_PageErase(uint32_t PageNum)	239
6.88.2.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	239
6.89	DMA	240
6.89.1	Подробное описание	240
6.90	Приватные данные	241
6.90.1	Подробное описание	241
6.91	Приватные константы	242
6.91.1	Подробное описание	242
6.92	Начальные значения регистров	243
6.93	Приватные функции	244
6.93.1	Подробное описание	245
6.93.2	Функции	245
6.93.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	245
6.93.2.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	245
6.93.2.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	245
6.93.2.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	246
6.93.2.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	246
6.93.2.6	DMA_ClearErrorStatus()	247
6.93.2.7	DMA_DeInit()	247
6.93.2.8	DMA_ErrorStatus()	247
6.93.2.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	247
6.93.2.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	247
6.93.2.11	DMA_MasterEnableCmd(FunctionalState State)	249
6.93.2.12	DMA_MasterEnableStatus()	249
6.93.2.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	249
6.93.2.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	250
6.93.2.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	250
6.93.2.16	DMA_StateStatus()	250
6.93.2.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	251

6.93.2.18 DMA_SWRequestCmd(uint32_t DMA_Channel)	252
6.93.2.19 DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	252
6.93.2.20 DMA_WaitOnReqStatus(uint32_t DMA_Channel)	252
6.94 EXTMEM	254
6.94.1 Подробное описание	254
6.95 Приватные данные	255
6.95.1 Подробное описание	255
6.96 Приватные константы	256
6.96.1 Подробное описание	256
6.97 Начальные значения регистров	257
6.97.1 Подробное описание	257
6.97.2 Макросы	257
6.97.2.1 EXT_MEM_CFG_Reset_Value	257
6.98 Приватные функции	258
6.98.1 Подробное описание	258
6.98.2 Функции	258
6.98.2.1 EXTMEM_DeInit()	258
6.98.2.2 EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	258
6.98.2.3 EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	259
6.99 GPIO	261
6.99.1 Подробное описание	261
6.100 Приватные данные	263
6.100.1 Подробное описание	263
6.101 Приватные константы	264
6.101.1 Подробное описание	264
6.102 Начальные значения регистров	265
6.102.1 Подробное описание	265
6.102.2 Макросы	265
6.102.2.1 GPIO_DATAOUT_Reset_Value	265
6.102.2.2 GPIO_GPIODEN0_Reset_Value	265
6.102.2.3 GPIO_GPIODEN1_Reset_Value	265
6.102.2.4 GPIO_GPIODEN2_Reset_Value	266
6.102.2.5 GPIO_GPIODEN3_Reset_Value	266
6.102.2.6 GPIO_GPIODCTLx_Reset_Value	266
6.102.2.7 GPIO_GPIODSCTLx_Reset_Value	266
6.102.2.8 GPIO_GPIOPCTLx_Reset_Value	266
6.102.2.9 GPIO_GPIOPUCTLx_Reset_Value	266
6.102.2.10 GPIO_GPIOQEx_Reset_Value	266
6.102.2.11 GPIO_GPIOQMx_Reset_Value	266
6.102.2.12 GPIO_GPIOQPx_Reset_Value	267

6.102.2.13	GPIO_GPIOSEx_Reset_Value	267
6.103	Маски портов	268
6.103.1	Подробное описание	268
6.103.2	Макросы	268
6.103.2.1	GPIO_Regs_A_C_E_G_Mask	268
6.103.2.2	GPIO_Regs_B_D_F_H_Mask	268
6.103.2.3	GPIO_Regs_GPIOA_Mask	268
6.103.2.4	GPIO_Regs_GPIOB_Mask	269
6.103.2.5	GPIO_Regs_GPIOC_Mask	269
6.103.2.6	GPIO_Regs_GPIOD_Mask	269
6.103.2.7	GPIO_Regs_GPIOE_Mask	269
6.103.2.8	GPIO_Regs_GPIOF_Mask	269
6.103.2.9	GPIO_Regs_GPIOG_Mask	269
6.103.2.10	GPIO_Regs_GPIOH_Mask	269
6.104	Приватные функции	270
6.104.1	Подробное описание	271
6.104.2	Функции	271
6.104.2.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	271
6.104.2.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	271
6.104.2.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	271
6.104.2.4	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	272
6.104.2.5	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	272
6.104.2.6	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	273
6.104.2.7	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	273
6.104.2.8	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	273
6.104.2.9	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	274
6.104.2.10	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	274
6.104.2.11	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	274
6.104.2.12	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	275
6.104.2.13	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	275
6.104.2.14	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	275
6.104.2.15	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	276
6.104.2.16	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	276
6.104.2.17	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	276

6.104.2.18	<code>GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)</code>	277
6.105	RCC	278
6.105.1	Подробное описание	278
6.106	Приватные данные	279
6.106.1	Подробное описание	279
6.107	Приватные константы	280
6.107.1	Подробное описание	280
6.108	Начальные значения регистров	281
6.108.1	Подробное описание	281
6.108.2	Макросы	281
6.108.2.1	<code>RCC_PLL_CTRL_Reset_Value</code>	281
6.108.2.2	<code>RCC_PLL_NF_Reset_Value</code>	281
6.108.2.3	<code>RCC_PLL_NR_Reset_Value</code>	281
6.108.2.4	<code>RCC_PLL_OD_Reset_Value</code>	281
6.109	Приватные функции	282
6.109.1	Подробное описание	283
6.109.2	Функции	283
6.109.2.1	<code>RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)</code>	283
6.109.2.2	<code>RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)</code>	283
6.109.2.3	<code>RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)</code>	284
6.109.2.4	<code>RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)</code>	284
6.109.2.5	<code>RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)</code>	285
6.109.2.6	<code>RCC_PLLDeInit()</code>	285
6.109.2.7	<code>RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)</code>	285
6.109.2.8	<code>RCC_PLLPowerDownCmd(FunctionalState State)</code>	286
6.109.2.9	<code>RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)</code>	286
6.109.2.10	<code>RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)</code>	287
6.109.2.11	<code>RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)</code>	287
6.109.2.12	<code>RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)</code>	287
6.109.2.13	<code>RCC_SysClkDiv2Out(FunctionalState State)</code>	288
6.109.2.14	<code>RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)</code>	288
6.109.2.15	<code>RCC_SysClkStatus()</code>	288
6.109.2.16	<code>RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)</code>	289

6.109.2.1	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	289
6.109.2.1	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	289
6.109.2.1	RCC_USBClkCmd(FunctionalState State)	290
6.109.2.2	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	290
6.109.2.2	RCC_WaitClkChange(RCC_SysClk_TypeDef RCC_SysClk)	290
6.110	RTC	291
6.110.1	Подробное описание	291
6.111	Приватные данные	292
6.111.1	Подробное описание	292
6.112	Приватные функции	293
6.112.1	Подробное описание	293
6.113	TIMER	294
6.113.1	Подробное описание	294
6.114	Приватные данные	295
6.114.1	Подробное описание	295
6.115	Приватные константы	296
6.116	Приватные функции	297
6.116.1	Подробное описание	297
6.116.2	Функции	297
6.116.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	297
6.116.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	298
6.116.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	298
6.116.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	298
6.116.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	299
6.116.2.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	299
6.116.2.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	299
6.116.2.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	299
6.116.2.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	300
6.116.2.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	300
6.116.2.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	300
6.117	UART	302
6.117.1	Подробное описание	302
6.118	Приватные данные	303
6.118.1	Подробное описание	303

6.119	Приватные константы	304
6.120	Приватные функции	305
6.120.1	Подробное описание	306
6.120.2	Функции	306
6.120.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	306
6.120.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	306
6.120.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	307
6.120.2.4	UART_DeInit(NT_UART_TypeDef *UARTx)	307
6.120.2.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	307
6.120.2.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	307
6.120.2.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error)	308
6.120.2.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	308
6.120.2.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag)	308
6.120.2.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	309
6.120.2.11	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)	309
6.120.2.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	309
6.120.2.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	310
6.120.2.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	310
6.120.2.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	310
6.120.2.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	311
6.120.2.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	311
6.120.2.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	311
6.120.2.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	312
6.120.2.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	312
6.121	USERFLASH	313
6.121.1	Подробное описание	313
6.122	Приватные данные	314
6.122.1	Подробное описание	314
6.123	Приватные константы	315
6.124	Приватные функции	316

6.124.1	Подробное описание	316
6.124.2	Функции	316
6.124.2.1	USERFLASH_FullErase()	316
6.124.2.2	USERFLASH_Info_PageErase(uint32_t PageNum)	317
6.124.2.3	USERFLASH_Info_Read(uint32_t Address)	317
6.124.2.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	317
6.124.2.5	USERFLASH_Init(uint32_t SysClkFreq)	317
6.124.2.6	USERFLASH_ITCmd(FunctionalState State)	318
6.124.2.7	USERFLASH_OperationStatus()	318
6.124.2.8	USERFLASH_OperationStatusClear()	318
6.124.2.9	USERFLASH_PageErase(uint32_t PageNum)	318
6.124.2.10	USERFLASH_Read(uint32_t Address)	319
6.124.2.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	319
6.125	Драйвер периферии	320
6.125.1	Подробное описание	320
6.126	Настройка драйвера	321
6.126.1	Подробное описание	321
6.126.2	Макросы	321
6.126.2.1	EXT_OSC_VALUE	321
6.126.2.2	INT_OSC_VALUE	321
6.127	Макросы	322
6.127.1	Подробное описание	322
6.128	Типы	323
6.128.1	Подробное описание	323
6.128.2	Макросы	323
6.128.2.1	IS_GPIO_ALL_PERIPH	323
6.128.2.2	IS_SPI_ALL_PERIPH	324
6.128.2.3	IS_TIMER_ALL_PERIPH	324
6.128.2.4	IS_UART_ALL_PERIPH	324
6.129	Периферия	325
6.129.1	Подробное описание	326
7	Структуры данных	327
7.1	Структура CHANNEL_CFG_bits	327
7.1.1	Подробное описание	327
7.1.2	Поля	327
7.1.2.1	CYCLE_CTRL	327
7.1.2.2	DST_INC	328
7.1.2.3	DST_PROT_BUFFERABLE	328
7.1.2.4	DST_PROT_CACHEABLE	328

7.1.2.5	DST_PROT_PRIVILEGED	328
7.1.2.6	DST_SIZE	328
7.1.2.7	N_MINUS_1	328
7.1.2.8	NEXT_USEBURST	328
7.1.2.9	R_POWER	328
7.1.2.10	SRC_INC	329
7.1.2.11	SRC_PROT_BUFFERABLE	329
7.1.2.12	SRC_PROT_CACHEABLE	329
7.1.2.13	SRC_PROT_PRIVILEGED	329
7.1.2.14	SRC_SIZE	329
7.2	Структура ADC_DC_Init_TypeDef	329
7.2.1	Подробное описание	330
7.2.2	Поля	330
7.2.2.1	ADC_DC_Channel	330
7.2.2.2	ADC_DC_Condition	330
7.2.2.3	ADC_DC_Mode	330
7.2.2.4	ADC_DC_ThresholdHigh	330
7.2.2.5	ADC_DC_ThresholdLow	330
7.3	Структура ADC_Init_TypeDef	331
7.3.1	Подробное описание	331
7.3.2	Поля	331
7.3.2.1	ADC_Average	331
7.3.2.2	ADC_Measure_A	331
7.3.2.3	ADC_Measure_B	331
7.3.2.4	ADC_Mode	332
7.3.2.5	ADC_Phase	332
7.3.2.6	ADC_Resolution	332
7.4	Структура ADC_SEQ_Init_TypeDef	332
7.4.1	Подробное описание	332
7.4.2	Поля	332
7.4.2.1	ADC_Channels	332
7.4.2.2	ADC_SEQ_ConversionCount	333
7.4.2.3	ADC_SEQ_ConversionDelay	333
7.4.2.4	ADC_SEQ_DC	333
7.4.2.5	ADC_SEQ_StartEvent	333
7.4.2.6	ADC_SEQ_SWReqEn	333
7.5	Структура DMA_Channel_TypeDef	333
7.5.1	Подробное описание	334
7.5.2	Поля	334
7.5.2.1	CHANNEL_CFG	334

7.5.2.2	CHANNEL_CFG_bit	334
7.5.2.3	DST_DATA_END	334
7.5.2.4	SRC_DATA_END	335
7.6	Структура DMA_ChannelInit_TypeDef	335
7.6.1	Подробное описание	336
7.6.2	Поля	336
7.6.2.1	DMA_ArbitrationRate	336
7.6.2.2	DMA_DstDataEndPtr	336
7.6.2.3	DMA_DstDataInc	336
7.6.2.4	DMA_DstDataSize	336
7.6.2.5	DMA_DstProtect	336
7.6.2.6	DMA_Mode	336
7.6.2.7	DMA_NextUseburst	337
7.6.2.8	DMA_SrcDataEndPtr	337
7.6.2.9	DMA_SrcDataInc	337
7.6.2.10	DMA_SrcDataSize	337
7.6.2.11	DMA_SrcProtect	337
7.6.2.12	DMA_TransfersTotal	337
7.7	Структура DMA_ConfigData_TypeDef	337
7.7.1	Подробное описание	338
7.7.2	Поля	338
7.7.2.1	ALT_DATA	338
7.7.2.2	PRM_DATA	339
7.7.2.3	RESERVED0	339
7.7.2.4	RESERVED1	339
7.8	Структура DMA_ConfigStruct_TypeDef	339
7.8.1	Подробное описание	340
7.8.2	Поля	340
7.8.2.1	CH	340
7.9	Структура DMA_Init_TypeDef	340
7.9.1	Подробное описание	340
7.9.2	Поля	341
7.9.2.1	DMA_Channel	341
7.9.2.2	DMA_ChannelEnable	341
7.9.2.3	DMA_HighPriority	341
7.9.2.4	DMA_PrmAlt	341
7.9.2.5	DMA_Protection	341
7.9.2.6	DMA_ReqMask	341
7.9.2.7	DMA_UseBurst	341
7.10	Структура DMA_Protect_TypeDef	342

7.10.1	Подробное описание	342
7.10.2	Поля	342
7.10.2.1	BUFFERABLE	342
7.10.2.2	CACHEABLE	342
7.10.2.3	PRIVELGED	342
7.11	Структура EXTMEM_Init_TypeDef	343
7.11.1	Подробное описание	343
7.11.2	Поля	343
7.11.2.1	CEMask	343
7.11.2.2	EXTMEM_ReadWaitState	343
7.11.2.3	EXTMEM_RWWaitState	343
7.11.2.4	EXTMEM_Width	343
7.11.2.5	EXTMEM_WriteWaitState	344
7.12	Структура GPIO_Init_TypeDef	344
7.12.1	Подробное описание	344
7.12.2	Поля	344
7.12.2.1	GPIO_AltFunc	344
7.12.2.2	GPIO_Dir	344
7.12.2.3	GPIO_Load	345
7.12.2.4	GPIO_Mode	345
7.12.2.5	GPIO_Out	345
7.12.2.6	GPIO_OutMode	345
7.12.2.7	GPIO_Pin	345
7.12.2.8	GPIO_PullUp	345
7.13	Структура RCC_PLLInit_TypeDef	345
7.13.1	Подробное описание	346
7.13.2	Поля	346
7.13.2.1	RCC_PLLDiv	346
7.13.2.2	RCC_PLLNF	346
7.13.2.3	RCC_PLLNO	346
7.13.2.4	RCC_PLLNR	346
7.13.2.5	RCC_PLLRef	346
7.14	Структура RTC_Date_TypeDef	347
7.14.1	Подробное описание	347
7.14.2	Поля	347
7.14.2.1	RTC_Day	347
7.14.2.2	RTC_Month	347
7.14.2.3	RTC_Weekday	347
7.14.2.4	RTC_Year	347
7.15	Структура RTC_Time_TypeDef	348

7.15.1	Подробное описание	348
7.15.2	Поля	348
7.15.2.1	RTC_Hour	348
7.15.2.2	RTC_Minute	348
7.15.2.3	RTC_Psecond	348
7.15.2.4	RTC_Second	348
7.16	Структура UART_Init_TypeDef	349
7.16.1	Подробное описание	349
7.16.2	Поля	349
7.16.2.1	UART_BaudRate	349
7.16.2.2	UART_ClkFreq	349
7.16.2.3	UART_DataWidth	349
7.16.2.4	UART_FIFOEn	349
7.16.2.5	UART_FIFOLevelRx	350
7.16.2.6	UART_FIFOLevelTx	350
7.16.2.7	UART_ParityBit	350
7.16.2.8	UART_RxEn	350
7.16.2.9	UART_StopBit	350
7.16.2.10	UART_TxEn	350
7.17	Структура UART_ModemInit_TypeDef	351
7.17.1	Подробное описание	351
7.17.2	Поля	351
7.17.2.1	UART_CTSEn	351
7.17.2.2	UART_InvDTR	351
7.17.2.3	UART_InvRTS	351
7.17.2.4	UART_RTSEn	351
8	Файлы	353
8.1	Файл niietcm4.h	353
8.1.1	Подробное описание	354
8.1.2	Макросы	355
8.1.2.1	EXT_OSC_VALUE	355
8.1.2.2	INT_OSC_VALUE	355
8.1.2.3	IS_GPIO_ALL_PERIPH	355
8.1.2.4	IS_SPI_ALL_PERIPH	356
8.1.2.5	IS_TIMER_ALL_PERIPH	356
8.1.2.6	IS_UART_ALL_PERIPH	356
8.2	Файл niietcm4_adc.c	356
8.2.1	Подробное описание	359
8.2.2	Функции	359

8.2.2.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	359
8.2.2.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	360
8.2.2.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module)	360
8.2.2.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	360
8.2.2.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↔ DC_DC_Init_TypeDef *ADC_DC_InitStruct)	361
8.2.2.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	361
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)	361
8.2.2.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	362
8.2.2.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	362
8.2.2.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	362
8.2.2.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	363
8.2.2.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	363
8.2.2.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	363
8.2.2.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	364
8.2.2.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	364
8.2.2.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	364
8.2.2.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↔ Def *ADC_InitStruct)	365
8.2.2.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	366
8.2.2.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	366
8.2.2.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State)	366
8.2.2.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ↔ _Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)	367
8.2.2.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↔ SEQ_Module)	367
8.2.2.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef A↔ DC_SEQ_Module)	367
8.2.2.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC↔ _SEQ_Module)	368
8.2.2.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	368

8.2.2.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	368
8.2.2.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	369
8.2.2.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	369
8.2.2.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	369
8.2.2.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	370
8.2.2.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	371
8.2.2.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	371
8.2.2.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	371
8.2.2.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	372
8.2.2.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	372
8.2.2.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	372
8.2.2.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	373
8.2.2.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	373
8.2.2.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	373
8.2.2.40	ADC_SEQ_SWReq()	374
8.2.2.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	374
8.3	Файл niietcm4_adc.h	374
8.3.1	Подробное описание	381
8.3.2	Макросы	381
8.3.2.1	ADC_Channel_0	381
8.3.2.2	ADC_Channel_1	381
8.3.2.3	ADC_Channel_10	381
8.3.2.4	ADC_Channel_11	382
8.3.2.5	ADC_Channel_12	382
8.3.2.6	ADC_Channel_13	382
8.3.2.7	ADC_Channel_14	382
8.3.2.8	ADC_Channel_15	382
8.3.2.9	ADC_Channel_16	382
8.3.2.10	ADC_Channel_17	382

8.3.2.11	ADC_Channel_18	382
8.3.2.12	ADC_Channel_19	382
8.3.2.13	ADC_Channel_2	382
8.3.2.14	ADC_Channel_20	383
8.3.2.15	ADC_Channel_21	383
8.3.2.16	ADC_Channel_22	383
8.3.2.17	ADC_Channel_23	383
8.3.2.18	ADC_Channel_3	383
8.3.2.19	ADC_Channel_4	383
8.3.2.20	ADC_Channel_5	383
8.3.2.21	ADC_Channel_6	383
8.3.2.22	ADC_Channel_7	383
8.3.2.23	ADC_Channel_8	384
8.3.2.24	ADC_Channel_9	384
8.3.2.25	ADC_Channel_All	384
8.3.2.26	ADC_Channel_None	384
8.3.2.27	ADC_DC_0	384
8.3.2.28	ADC_DC_1	384
8.3.2.29	ADC_DC_10	384
8.3.2.30	ADC_DC_11	384
8.3.2.31	ADC_DC_12	384
8.3.2.32	ADC_DC_13	385
8.3.2.33	ADC_DC_14	385
8.3.2.34	ADC_DC_15	385
8.3.2.35	ADC_DC_16	385
8.3.2.36	ADC_DC_17	385
8.3.2.37	ADC_DC_18	385
8.3.2.38	ADC_DC_19	385
8.3.2.39	ADC_DC_2	385
8.3.2.40	ADC_DC_20	385
8.3.2.41	ADC_DC_21	385
8.3.2.42	ADC_DC_22	386
8.3.2.43	ADC_DC_23	386
8.3.2.44	ADC_DC_3	386
8.3.2.45	ADC_DC_4	386
8.3.2.46	ADC_DC_5	386
8.3.2.47	ADC_DC_6	386
8.3.2.48	ADC_DC_7	386
8.3.2.49	ADC_DC_8	386
8.3.2.50	ADC_DC_9	386

8.3.2.51	ADC_DC_All	387
8.3.2.52	ADC_DC_None	387
8.3.2.53	ADC_SEQ_0	387
8.3.2.54	ADC_SEQ_1	387
8.3.2.55	ADC_SEQ_2	387
8.3.2.56	ADC_SEQ_3	387
8.3.2.57	ADC_SEQ_4	387
8.3.2.58	ADC_SEQ_5	387
8.3.2.59	ADC_SEQ_6	387
8.3.2.60	ADC_SEQ_7	388
8.3.2.61	IS_ADC_AVERAGE	388
8.3.2.62	IS_ADC_DC_CHANNEL	388
8.3.2.63	IS_ADC_DC_CONDITION	388
8.3.2.64	IS_ADC_DC_MODE	389
8.3.2.65	IS_ADC_DC_MODULE	389
8.3.2.66	IS_ADC_MEASURE	389
8.3.2.67	IS_ADC_MODE	390
8.3.2.68	IS_ADC_MODULE	390
8.3.2.69	IS_ADC_RESOLUTION	390
8.3.2.70	IS_ADC_SEQ_FIFO_LEVEL	390
8.3.2.71	IS_ADC_SEQ_MODULE	391
8.3.2.72	IS_ADC_SEQ_START_EVENT	391
8.3.3	Перечисления	391
8.3.3.1	ADC_Average_TypeDef	391
8.3.3.2	ADC_DC_Channel_TypeDef	392
8.3.3.3	ADC_DC_Condition_TypeDef	393
8.3.3.4	ADC_DC_Mode_TypeDef	393
8.3.3.5	ADC_DC_Module_TypeDef	393
8.3.3.6	ADC_Measure_TypeDef	394
8.3.3.7	ADC_Mode_TypeDef	394
8.3.3.8	ADC_Module_TypeDef	394
8.3.3.9	ADC_Resolution_TypeDef	395
8.3.3.10	ADC_SEQ_FIFOLevel_TypeDef	395
8.3.3.11	ADC_SEQ_Module_TypeDef	395
8.3.3.12	ADC_SEQ_StartEvent_TypeDef	395
8.3.4	Функции	396
8.3.4.1	ADC_Cmd(ADC_Module_TypeDef ADC_Module, FunctionalState State)	396
8.3.4.2	ADC_DC_Cmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	396

8.3.4.3	ADC_DC_DeInit(ADC_DC_Module_TypeDef ADC_DC_Module) . .	396
8.3.4.4	ADC_DC_GetLastData(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	397
8.3.4.5	ADC_DC_Init(ADC_DC_Module_TypeDef ADC_DC_Module, A↔ DC_DC_Init_TypeDef *ADC_DC_InitStruct)	397
8.3.4.6	ADC_DC_ITCmd(ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)	397
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)	398
8.3.4.8	ADC_DC_ITGenCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	398
8.3.4.9	ADC_DC_ITMaskCmd(ADC_DC_Module_TypeDef ADC_DC_↔ Module, FunctionalState State)	399
8.3.4.10	ADC_DC_ITMaskedStatus(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	400
8.3.4.11	ADC_DC_ITRawStatus(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	400
8.3.4.12	ADC_DC_ITStatusClear(ADC_DC_Module_TypeDef ADC_DC_↔ Module)	400
8.3.4.13	ADC_DC_StructInit(ADC_DC_Init_TypeDef *ADC_DC_InitStruct)	401
8.3.4.14	ADC_DC_TrigStatus(ADC_DC_Module_TypeDef ADC_DC_Module)	401
8.3.4.15	ADC_DC_TrigStatusClear(ADC_DC_Module_TypeDef ADC_DC↔ _Module)	401
8.3.4.16	ADC_DeInit(ADC_Module_TypeDef ADC_Module)	402
8.3.4.17	ADC_Init(ADC_Module_TypeDef ADC_Module, ADC_Init_Type↔ Def *ADC_InitStruct)	402
8.3.4.18	ADC_SEQ_Cmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	402
8.3.4.19	ADC_SEQ_DeInit(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	403
8.3.4.20	ADC_SEQ_DMAMCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_↔ Module, FunctionalState State)	403
8.3.4.21	ADC_SEQ_DMAConfig(ADC_SEQ_Module_TypeDef ADC_SEQ↔ _Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel) .	403
8.3.4.22	ADC_SEQ_DMAErrorStatus(ADC_SEQ_Module_TypeDef ADC_↔ SEQ_Module)	404
8.3.4.23	ADC_SEQ_DMAErrorStatusClear(ADC_SEQ_Module_TypeDef A↔ DC_SEQ_Module)	404
8.3.4.24	ADC_SEQ_FIFOEmptyStatus(ADC_SEQ_Module_TypeDef ADC↔ _SEQ_Module)	404
8.3.4.25	ADC_SEQ_FIFOEmptyStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	405
8.3.4.26	ADC_SEQ_FIFOFullStatus(ADC_SEQ_Module_TypeDef ADC_S↔ EQ_Module)	406
8.3.4.27	ADC_SEQ_FIFOFullStatusClear(ADC_SEQ_Module_TypeDef AD↔ C_SEQ_Module)	406

8.3.4.28	ADC_SEQ_GetConversionCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	406
8.3.4.29	ADC_SEQ_GetFIFOData(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	407
8.3.4.30	ADC_SEQ_GetFIFOLoad(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	407
8.3.4.31	ADC_SEQ_GetITCount(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	407
8.3.4.32	ADC_SEQ_Init(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	408
8.3.4.33	ADC_SEQ_ITCmd(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)	409
8.3.4.34	ADC_SEQ_ITConfig(ADC_SEQ_Module_TypeDef ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst)	409
8.3.4.35	ADC_SEQ_ITCountRst(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	410
8.3.4.36	ADC_SEQ_ITMaskedStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	410
8.3.4.37	ADC_SEQ_ITRawStatus(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	410
8.3.4.38	ADC_SEQ_ITStatusClear(ADC_SEQ_Module_TypeDef ADC_SEQ_Module)	410
8.3.4.39	ADC_SEQ_StructInit(ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)	411
8.3.4.40	ADC_SEQ_SWReq()	411
8.3.4.41	ADC_StructInit(ADC_Init_TypeDef *ADC_InitStruct)	411
8.4	Файл niietcm4_bootflash.c	411
8.4.1	Подробное описание	412
8.4.2	Функции	413
8.4.2.1	BOOTFLASH_FullErase()	413
8.4.2.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	413
8.4.2.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	413
8.4.2.4	BOOTFLASH_Init(uint32_t SysClkFreq)	414
8.4.2.5	BOOTFLASH_ITCmd(FunctionalState State)	414
8.4.2.6	BOOTFLASH_OperationStatus()	414
8.4.2.7	BOOTFLASH_OperationStatusClear()	414
8.4.2.8	BOOTFLASH_PageErase(uint32_t PageNum)	415
8.4.2.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	415
8.5	Файл niietcm4_bootflash.h	415
8.5.1	Подробное описание	417
8.5.2	Макросы	417

8.5.2.1	BOOTFLASH_INFO_PAGE_SIZE_BYTES	417
8.5.2.2	BOOTFLASH_INFO_PAGE_TOTAL	418
8.5.2.3	BOOTFLASH_INFO_TOTAL_BYTES	418
8.5.2.4	BOOTFLASH_PAGE_SIZE_BYTES	418
8.5.2.5	BOOTFLASH_PAGE_TOTAL	418
8.5.2.6	BOOTFLASH_TOTAL_BYTES	418
8.5.2.7	IS_BOOTFLASH_STATUS	418
8.5.3	Перечисления	418
8.5.3.1	BOOTFLASH_Status_TypeDef	418
8.5.4	Функции	419
8.5.4.1	BOOTFLASH_FullErase()	419
8.5.4.2	BOOTFLASH_Info_PageErase(uint32_t PageNum)	419
8.5.4.3	BOOTFLASH_Info_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	419
8.5.4.4	BOOTFLASH_Init(uint32_t SysClkFreq)	419
8.5.4.5	BOOTFLASH_ITCmd(FunctionalState State)	420
8.5.4.6	BOOTFLASH_OperationStatus()	420
8.5.4.7	BOOTFLASH_OperationStatusClear()	420
8.5.4.8	BOOTFLASH_PageErase(uint32_t PageNum)	420
8.5.4.9	BOOTFLASH_Write(uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)	421
8.6	Файл niietcm4_conf.h	422
8.6.1	Подробное описание	422
8.7	Файл niietcm4_dma.c	423
8.7.1	Подробное описание	424
8.7.2	Функции	425
8.7.2.1	DMA_BasePtrConfig(uint32_t BasePtr)	425
8.7.2.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	425
8.7.2.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	425
8.7.2.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	426
8.7.2.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	426
8.7.2.6	DMA_ClearErrorStatus()	427
8.7.2.7	DMA_DeInit()	427
8.7.2.8	DMA_ErrorStatus()	427
8.7.2.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	427
8.7.2.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	428
8.7.2.11	DMA_MasterEnableCmd(FunctionalState State)	428
8.7.2.12	DMA_MasterEnableStatus()	428

8.7.2.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	429
8.7.2.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	429
8.7.2.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	429
8.7.2.16	DMA_StateStatus()	430
8.7.2.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	430
8.7.2.18	DMA_SWRequestCmd(uint32_t DMA_Channel)	430
8.7.2.19	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	430
8.7.2.20	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	431
8.8	Файл niietcm4_dma.h	431
8.8.1	Подробное описание	435
8.8.2	Макросы	436
8.8.2.1	CHANNEL_CFG_CYCLE_CTRL_Msk	436
8.8.2.2	CHANNEL_CFG_CYCLE_CTRL_Pos	436
8.8.2.3	CHANNEL_CFG_DST_INC_Msk	436
8.8.2.4	CHANNEL_CFG_DST_INC_Pos	436
8.8.2.5	CHANNEL_CFG_DST_PROT_CTRL_Msk	436
8.8.2.6	CHANNEL_CFG_DST_PROT_CTRL_Pos	436
8.8.2.7	CHANNEL_CFG_DST_SIZE_Msk	437
8.8.2.8	CHANNEL_CFG_DST_SIZE_Pos	437
8.8.2.9	CHANNEL_CFG_N_MINUS_1_Msk	437
8.8.2.10	CHANNEL_CFG_N_MINUS_1_Pos	437
8.8.2.11	CHANNEL_CFG_NEXT_USEBURST_Msk	437
8.8.2.12	CHANNEL_CFG_NEXT_USEBURST_Pos	437
8.8.2.13	CHANNEL_CFG_R_POWER_Msk	437
8.8.2.14	CHANNEL_CFG_R_POWER_Pos	437
8.8.2.15	CHANNEL_CFG_SRC_INC_Msk	437
8.8.2.16	CHANNEL_CFG_SRC_INC_Pos	438
8.8.2.17	CHANNEL_CFG_SRC_PROT_CTRL_Msk	438
8.8.2.18	CHANNEL_CFG_SRC_PROT_CTRL_Pos	438
8.8.2.19	CHANNEL_CFG_SRC_SIZE_Msk	438
8.8.2.20	CHANNEL_CFG_SRC_SIZE_Pos	438
8.8.2.21	DMA_Channel_0	438
8.8.2.22	DMA_Channel_1	438
8.8.2.23	DMA_Channel_10	438
8.8.2.24	DMA_Channel_11	438
8.8.2.25	DMA_Channel_12	438
8.8.2.26	DMA_Channel_13	439
8.8.2.27	DMA_Channel_14	439
8.8.2.28	DMA_Channel_15	439
8.8.2.29	DMA_Channel_16	439

8.8.2.30	DMA_Channel_17	439
8.8.2.31	DMA_Channel_18	439
8.8.2.32	DMA_Channel_19	439
8.8.2.33	DMA_Channel_2	439
8.8.2.34	DMA_Channel_20	439
8.8.2.35	DMA_Channel_21	440
8.8.2.36	DMA_Channel_22	440
8.8.2.37	DMA_Channel_23	440
8.8.2.38	DMA_Channel_3	440
8.8.2.39	DMA_Channel_4	440
8.8.2.40	DMA_Channel_5	440
8.8.2.41	DMA_Channel_6	440
8.8.2.42	DMA_Channel_7	440
8.8.2.43	DMA_Channel_8	440
8.8.2.44	DMA_Channel_9	440
8.8.2.45	DMA_Channel_ADCSEQ0	441
8.8.2.46	DMA_Channel_ADCSEQ1	441
8.8.2.47	DMA_Channel_ADCSEQ2	441
8.8.2.48	DMA_Channel_ADCSEQ3	441
8.8.2.49	DMA_Channel_ADCSEQ4	441
8.8.2.50	DMA_Channel_ADCSEQ5	441
8.8.2.51	DMA_Channel_ADCSEQ6	441
8.8.2.52	DMA_Channel_ADCSEQ7	441
8.8.2.53	DMA_Channel_All	441
8.8.2.54	DMA_Channel_SPI0_RX	442
8.8.2.55	DMA_Channel_SPI0_TX	442
8.8.2.56	DMA_Channel_SPI1_RX	442
8.8.2.57	DMA_Channel_SPI1_TX	442
8.8.2.58	DMA_Channel_SPI2_RX	442
8.8.2.59	DMA_Channel_SPI2_TX	442
8.8.2.60	DMA_Channel_SPI3_RX	442
8.8.2.61	DMA_Channel_SPI3_TX	442
8.8.2.62	DMA_Channel_UART0_RX	442
8.8.2.63	DMA_Channel_UART0_TX	442
8.8.2.64	DMA_Channel_UART1_RX	443
8.8.2.65	DMA_Channel_UART1_TX	443
8.8.2.66	DMA_Channel_UART2_RX	443
8.8.2.67	DMA_Channel_UART2_TX	443
8.8.2.68	DMA_Channel_UART3_RX	443
8.8.2.69	DMA_Channel_UART3_TX	443

8.8.2.70	IS_DMA_ARBITRATION_RATE	443
8.8.2.71	IS_DMA_DATA_INC	444
8.8.2.72	IS_DMA_DATA_SIZE	444
8.8.2.73	IS_DMA_MODE	444
8.8.2.74	IS_DMA_STATE	444
8.8.2.75	IS_GET_DMA_CHANNEL	445
8.8.3	Перечисления	445
8.8.3.1	DMA_ArbitrationRate_TypeDef	445
8.8.3.2	DMA_DataInc_TypeDef	446
8.8.3.3	DMA_DataSize_TypeDef	446
8.8.3.4	DMA_Mode_TypeDef	446
8.8.3.5	DMA_State_TypeDef	447
8.8.4	Функции	447
8.8.4.1	DMA_BasePtrConfig(uint32_t BasePtr)	447
8.8.4.2	DMA_ChannelDeInit(DMA_Channel_TypeDef *DMA_Channel)	447
8.8.4.3	DMA_ChannelEnableCmd(uint32_t DMA_Channel, FunctionalState State)	447
8.8.4.4	DMA_ChannelInit(DMA_Channel_TypeDef *DMA_Channel, DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	448
8.8.4.5	DMA_ChannelStructInit(DMA_ChannelInit_TypeDef *DMA_ChannelInitStruct)	448
8.8.4.6	DMA_ClearErrorStatus()	449
8.8.4.7	DMA_DeInit()	449
8.8.4.8	DMA_ErrorStatus()	449
8.8.4.9	DMA_HighPriorityCmd(uint32_t DMA_Channel, FunctionalState State)	449
8.8.4.10	DMA_Init(DMA_Init_TypeDef *DMA_InitStruct)	450
8.8.4.11	DMA_MasterEnableCmd(FunctionalState State)	450
8.8.4.12	DMA_MasterEnableStatus()	450
8.8.4.13	DMA_PrmAltCmd(uint32_t DMA_Channel, FunctionalState State)	451
8.8.4.14	DMA_ProtectionConfig(DMA_Protect_TypeDef *DMA_Protection)	451
8.8.4.15	DMA_ReqMaskCmd(uint32_t DMA_Channel, FunctionalState State)	451
8.8.4.16	DMA_StateStatus()	452
8.8.4.17	DMA_StructInit(DMA_Init_TypeDef *DMA_InitStruct)	452
8.8.4.18	DMA_SWRequestCmd(uint32_t DMA_Channel)	452
8.8.4.19	DMA_UseBurstCmd(uint32_t DMA_Channel, FunctionalState State)	452
8.8.4.20	DMA_WaitOnReqStatus(uint32_t DMA_Channel)	453
8.9	Файл niietcm4_extmem.c	453
8.9.1	Подробное описание	454
8.9.2	Макросы	454
8.9.2.1	EXT_MEM_CFG_Reset_Value	454
8.9.3	Функции	454

8.9.3.1	EXTMEM_DeInit()	454
8.9.3.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	455
8.9.3.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	455
8.10	Файл nnetcm4_extmem.h	455
8.10.1	Подробное описание	457
8.10.2	Макросы	458
8.10.2.1	EXTMEM_CEMask_Addr_11	458
8.10.2.2	EXTMEM_CEMask_Addr_11_19	458
8.10.2.3	EXTMEM_CEMask_Addr_12	458
8.10.2.4	EXTMEM_CEMask_Addr_13	458
8.10.2.5	EXTMEM_CEMask_Addr_14	458
8.10.2.6	EXTMEM_CEMask_Addr_15	458
8.10.2.7	EXTMEM_CEMask_Addr_16	458
8.10.2.8	EXTMEM_CEMask_Addr_17	458
8.10.2.9	EXTMEM_CEMask_Addr_18	459
8.10.2.10	EXTMEM_CEMask_Addr_19	459
8.10.2.11	IS_EXTMEM_READ_WAITSTATE	459
8.10.2.12	IS_EXTMEM_RW_WAITSTATE	459
8.10.2.13	IS_EXTMEM_WIDTH	459
8.10.2.14	IS_EXTMEM_WRITE_WAITSTATE	460
8.10.3	Перечисления	460
8.10.3.1	EXTMEM_ReadWaitState_TypeDef	460
8.10.3.2	EXTMEM_RWWaitState_TypeDef	460
8.10.3.3	EXTMEM_Width_TypeDef	461
8.10.3.4	EXTMEM_WriteWaitState_TypeDef	461
8.10.4	Функции	461
8.10.4.1	EXTMEM_DeInit()	461
8.10.4.2	EXTMEM_Init(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	461
8.10.4.3	EXTMEM_StructInit(EXTMEM_Init_TypeDef *EXTMEM_InitStruct)	462
8.11	Файл nnetcm4_gpio.c	463
8.11.1	Подробное описание	465
8.11.2	Макросы	465
8.11.2.1	GPIO_DATAOUT_Reset_Value	465
8.11.2.2	GPIO_GPIODEN0_Reset_Value	465
8.11.2.3	GPIO_GPIODEN1_Reset_Value	465
8.11.2.4	GPIO_GPIODEN2_Reset_Value	466
8.11.2.5	GPIO_GPIODEN3_Reset_Value	466
8.11.2.6	GPIO_GPIODCTLx_Reset_Value	466
8.11.2.7	GPIO_GPIODSCTLx_Reset_Value	466
8.11.2.8	GPIO_GPIOPCTLx_Reset_Value	466

8.11.2.9	GPIO_GPIOPUCTLx_Reset_Value	466
8.11.2.10	GPIO_GPIOQEx_Reset_Value	466
8.11.2.11	GPIO_GPIOQMx_Reset_Value	466
8.11.2.12	GPIO_GPIOQPx_Reset_Value	467
8.11.2.13	GPIO_GPIOSEx_Reset_Value	467
8.11.2.14	GPIO_Regs_A_C_E_G_Mask	467
8.11.2.15	GPIO_Regs_B_D_F_H_Mask	467
8.11.2.16	GPIO_Regs_GPIOA_Mask	467
8.11.2.17	GPIO_Regs_GPIOB_Mask	467
8.11.2.18	GPIO_Regs_GPIOC_Mask	467
8.11.2.19	GPIO_Regs_GPIOD_Mask	467
8.11.2.20	GPIO_Regs_GPIOE_Mask	467
8.11.2.21	GPIO_Regs_GPIOF_Mask	468
8.11.2.22	GPIO_Regs_GPIOG_Mask	468
8.11.2.23	GPIO_Regs_GPIOH_Mask	468
8.11.3	Функции	468
8.11.3.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	468
8.11.3.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	468
8.11.3.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	469
8.11.3.4	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	470
8.11.3.5	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	470
8.11.3.6	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	471
8.11.3.7	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	471
8.11.3.8	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	471
8.11.3.9	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	472
8.11.3.10	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	472
8.11.3.11	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	472
8.11.3.12	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	473
8.11.3.13	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	473
8.11.3.14	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	473
8.11.3.15	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	474
8.11.3.16	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	474
8.11.3.17	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	474
8.11.3.18	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	475

8.12 Файл <code>niietcm4_gpio.h</code>	475
8.12.1 Подробное описание	478
8.12.2 Макросы	479
8.12.2.1 <code>GPIO_Pin_0</code>	479
8.12.2.2 <code>GPIO_Pin_0_3</code>	479
8.12.2.3 <code>GPIO_Pin_0_7</code>	479
8.12.2.4 <code>GPIO_Pin_1</code>	479
8.12.2.5 <code>GPIO_Pin_10</code>	479
8.12.2.6 <code>GPIO_Pin_11</code>	479
8.12.2.7 <code>GPIO_Pin_12</code>	480
8.12.2.8 <code>GPIO_Pin_12_15</code>	480
8.12.2.9 <code>GPIO_Pin_13</code>	480
8.12.2.10 <code>GPIO_Pin_14</code>	480
8.12.2.11 <code>GPIO_Pin_15</code>	480
8.12.2.12 <code>GPIO_Pin_2</code>	480
8.12.2.13 <code>GPIO_Pin_3</code>	480
8.12.2.14 <code>GPIO_Pin_4</code>	480
8.12.2.15 <code>GPIO_Pin_4_7</code>	480
8.12.2.16 <code>GPIO_Pin_5</code>	480
8.12.2.17 <code>GPIO_Pin_6</code>	481
8.12.2.18 <code>GPIO_Pin_7</code>	481
8.12.2.19 <code>GPIO_Pin_8</code>	481
8.12.2.20 <code>GPIO_Pin_8_11</code>	481
8.12.2.21 <code>GPIO_Pin_8_15</code>	481
8.12.2.22 <code>GPIO_Pin_9</code>	481
8.12.2.23 <code>GPIO_Pin_All</code>	481
8.12.2.24 <code>IS_GET_GPIO_PIN</code>	481
8.12.2.25 <code>IS_GPIO_ALT_FUNC</code>	482
8.12.2.26 <code>IS_GPIO_DIR</code>	482
8.12.2.27 <code>IS_GPIO_INT_POL</code>	482
8.12.2.28 <code>IS_GPIO_INT_TYPE</code>	482
8.12.2.29 <code>IS_GPIO_LOAD</code>	483
8.12.2.30 <code>IS_GPIO_MODE</code>	483
8.12.2.31 <code>IS_GPIO_OUT</code>	483
8.12.2.32 <code>IS_GPIO_OUT_MODE</code>	483
8.12.2.33 <code>IS_GPIO_PULLUP</code>	483
8.12.2.34 <code>IS_GPIO_QUAL</code>	484
8.12.2.35 <code>IS_GPIO_QUAL_MODE</code>	484
8.12.2.36 <code>IS_GPIO_SYNC</code>	484
8.12.3 Перечисления	484

8.12.3.1	BitAction	484
8.12.3.2	GPIO_AltFunc_TypeDef	484
8.12.3.3	GPIO_Dir_TypeDef	485
8.12.3.4	GPIO_IntPol_TypeDef	485
8.12.3.5	GPIO_IntType_TypeDef	485
8.12.3.6	GPIO_Load_TypeDef	485
8.12.3.7	GPIO_Mode_TypeDef	485
8.12.3.8	GPIO_Out_TypeDef	486
8.12.3.9	GPIO_OutMode_TypeDef	486
8.12.3.10	GPIO_PullUp_TypeDef	486
8.12.3.11	GPIO_Qual_TypeDef	486
8.12.3.12	GPIO_QualMode_TypeDef	486
8.12.3.13	GPIO_Sync_TypeDef	487
8.12.4	Функции	487
8.12.4.1	GPIO_ClearBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	487
8.12.4.2	GPIO_DeInit(NT_GPIO_TypeDef *GPIOx)	487
8.12.4.3	GPIO_Init(NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)	487
8.12.4.4	GPIO_ITCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	488
8.12.4.5	GPIO_ITConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)	488
8.12.4.6	GPIO_ITStatusClear(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	489
8.12.4.7	GPIO_QualCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	489
8.12.4.8	GPIO_QualConfig(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)	489
8.12.4.9	GPIO_Read(NT_GPIO_TypeDef *GPIOx)	490
8.12.4.10	GPIO_ReadBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	490
8.12.4.11	GPIO_ReadMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)	490
8.12.4.12	GPIO_SetBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	491
8.12.4.13	GPIO_StructInit(GPIO_Init_TypeDef *GPIO_InitStruct)	491
8.12.4.14	GPIO_SyncCmd(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)	491
8.12.4.15	GPIO_ToggleBits(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)	492
8.12.4.16	GPIO_Write(NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)	492
8.12.4.17	GPIO_WriteBit(NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)	492
8.12.4.18	GPIO_WriteMask(NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)	493
8.13	Файл niietcm4_rcc.c	493
8.13.1	Подробное описание	495

8.13.2	Макросы	495
8.13.2.1	RCC_PLL_CTRL_Reset_Value	495
8.13.2.2	RCC_PLL_NF_Reset_Value	495
8.13.2.3	RCC_PLL_NR_Reset_Value	495
8.13.2.4	RCC_PLL_OD_Reset_Value	496
8.13.3	Функции	496
8.13.3.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	496
8.13.3.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	496
8.13.3.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	496
8.13.3.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	497
8.13.3.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	497
8.13.3.6	RCC_PLLDeInit()	498
8.13.3.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	498
8.13.3.8	RCC_PLLPowerDownCmd(FunctionalState State)	499
8.13.3.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	499
8.13.3.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	500
8.13.3.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	501
8.13.3.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	501
8.13.3.13	RCC_SysClkDiv2Out(FunctionalState State)	501
8.13.3.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	502
8.13.3.15	RCC_SysClkStatus()	502
8.13.3.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	502
8.13.3.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	503
8.13.3.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	503
8.13.3.19	RCC_USBClkCmd(FunctionalState State)	503
8.13.3.20	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	504
8.13.3.21	RCC_WaitClkChange(RCC_SysClk_TypeDef RCC_SysClk)	504
8.14	Файл niyetcm4_rcc.h	504
8.14.1	Подробное описание	508
8.14.2	Макросы	508
8.14.2.1	IS_RCC_ADC_CLK	508
8.14.2.2	IS_RCC_PERIPH_CLK	509

8.14.2.3	IS_RCC_PLL_NO	509
8.14.2.4	IS_RCC_PLL_REF	509
8.14.2.5	IS_RCC_SPI_CLK	510
8.14.2.6	IS_RCC_SYS_CLK	510
8.14.2.7	IS_RCC_UART_CLK	510
8.14.2.8	IS_RCC_USB_CLK	510
8.14.2.9	IS_RCC_USB_FREQ	511
8.14.2.10	RCC_CLK_CHANGE_TIMEOUT	511
8.14.3	Перечисления	511
8.14.3.1	RCC_ADCClk_TypeDef	511
8.14.3.2	RCC_PeriphClk_TypeDef	511
8.14.3.3	RCC_PeriphRst_TypeDef	512
8.14.3.4	RCC_PLLNO_TypeDef	513
8.14.3.5	RCC_PLLRef_TypeDef	513
8.14.3.6	RCC_SPIClk_TypeDef	513
8.14.3.7	RCC_SysClk_TypeDef	514
8.14.3.8	RCC_UARTClk_TypeDef	514
8.14.3.9	RCC_USBClk_TypeDef	514
8.14.3.10	RCC_USBFreq_TypeDef	514
8.14.4	Функции	515
8.14.4.1	RCC_ADCClkCmd(RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)	515
8.14.4.2	RCC_ADCClkDivConfig(RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)	516
8.14.4.3	RCC_PeriphClkCmd(RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)	516
8.14.4.4	RCC_PeriphRstCmd(RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)	517
8.14.4.5	RCC_PLLAutoConfig(RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)	517
8.14.4.6	RCC_PLLDeInit()	518
8.14.4.7	RCC_PLLInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	518
8.14.4.8	RCC_PLLPowerDownCmd(FunctionalState State)	519
8.14.4.9	RCC_PLLStructInit(RCC_PLLInit_TypeDef *RCC_PLLInit_Struct)	519
8.14.4.10	RCC_SPIClkCmd(NT_SPI_TypeDef *SPIx, FunctionalState State)	519
8.14.4.11	RCC_SPIClkDivConfig(NT_SPI_TypeDef *SPIx, uint32_t DivVal, FunctionalState DivState)	520
8.14.4.12	RCC_SPIClkSel(NT_SPI_TypeDef *SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)	520
8.14.4.13	RCC_SysClkDiv2Out(FunctionalState State)	520
8.14.4.14	RCC_SysClkSel(RCC_SysClk_TypeDef RCC_SysClk)	521
8.14.4.15	RCC_SysClkStatus()	521

8.14.4.16	RCC_UARTClkCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	521
8.14.4.17	RCC_UARTClkDivConfig(NT_UART_TypeDef *UARTx, uint32_t DivVal, FunctionalState DivState)	521
8.14.4.18	RCC_UARTClkSel(NT_UART_TypeDef *UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)	522
8.14.4.19	RCC_USBClkCmd(FunctionalState State)	522
8.14.4.20	RCC_USBClkConfig(RCC_USBClk_TypeDef RCC_USBClk, RCC_USBFreq_TypeDef RCC_USBFreq)	522
8.15	Файл niietcm4_rtc.c	523
8.15.1	Подробное описание	523
8.16	Файл niietcm4_rtc.h	524
8.16.1	Подробное описание	526
8.16.2	Макросы	526
8.16.2.1	IS_RTC_FORMAT	526
8.16.2.2	IS_RTC_MONTH	526
8.16.2.3	IS_RTC_WEEKDAY	527
8.16.3	Перечисления	527
8.16.3.1	RTC_Format_TypeDef	527
8.16.3.2	RTC_Month_TypeDef	527
8.16.3.3	RTC_Weekday_TypeDef	528
8.17	Файл niietcm4_timer.c	528
8.17.1	Подробное описание	529
8.17.2	Функции	529
8.17.2.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	529
8.17.2.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	530
8.17.2.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	530
8.17.2.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	530
8.17.2.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	531
8.17.2.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	531
8.17.2.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	531
8.17.2.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	531
8.17.2.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	532
8.17.2.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	532
8.17.2.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	532
8.18	Файл niietcm4_timer.h	533
8.18.1	Подробное описание	534

8.18.2	Макросы	535
8.18.2.1	IS_TIMER_EXT_INPUT	535
8.18.3	Перечисления	535
8.18.3.1	TIMER_ExtInput_TypeDef	535
8.18.4	Функции	535
8.18.4.1	TIMER_Cmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	535
8.18.4.2	TIMER_ExtInputConfig(NT_TIMER_TypeDef *TIMERx, TIMER_ExtInput_TypeDef TIMER_ExtInput)	535
8.18.4.3	TIMER_FreqConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerFreq)	536
8.18.4.4	TIMER_GetCounter(NT_TIMER_TypeDef *TIMERx)	536
8.18.4.5	TIMER_GetReload(NT_TIMER_TypeDef *TIMERx)	536
8.18.4.6	TIMER_ITCmd(NT_TIMER_TypeDef *TIMERx, FunctionalState State)	537
8.18.4.7	TIMER_ITStatus(NT_TIMER_TypeDef *TIMERx)	537
8.18.4.8	TIMER_ITStatusClear(NT_TIMER_TypeDef *TIMERx)	537
8.18.4.9	TIMER_PeriodConfig(NT_TIMER_TypeDef *TIMERx, uint32_t TimerClkFreq, uint32_t TimerPeriod)	538
8.18.4.10	TIMER_SetCounter(NT_TIMER_TypeDef *TIMERx, uint32_t CounterVal)	538
8.18.4.11	TIMER_SetReload(NT_TIMER_TypeDef *TIMERx, uint32_t ReloadVal)	538
8.19	Файл niietcm4_uart.c	538
8.19.1	Подробное описание	540
8.19.2	Функции	541
8.19.2.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t FracDiv)	541
8.19.2.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State)	542
8.19.2.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State)	542
8.19.2.4	UART_DeInit(NT_UART_TypeDef *UARTx)	542
8.19.2.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	543
8.19.2.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	543
8.19.2.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error)	543
8.19.2.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	544
8.19.2.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag)	545
8.19.2.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	545
8.19.2.11	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)	545

8.19.2.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_↵ Dir_Typedef UART_Dir, UART_FIFOLevel_TypeDef UART_FIF↵ OLevel)	546
8.19.2.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_IT↵ Source_Typedef UART_ITSource)	546
8.19.2.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_IT↵ Source_Typedef UART_ITSource)	546
8.19.2.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_IT↵ Source_Typedef UART_ITSource)	547
8.19.2.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_↵ ModemInit_TypeDef *UART_ModemInitStruct)	547
8.19.2.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_↵ ModemInitStruct)	547
8.19.2.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	548
8.19.2.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data) . . .	548
8.19.2.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	548
8.20	Файл niietcm4_uart.h	549
8.20.1	Подробное описание	552
8.20.2	Макросы	552
8.20.2.1	IS_UART_DATA_WIDTH	552
8.20.2.2	IS_UART_DIR	553
8.20.2.3	IS_UART_ERROR	553
8.20.2.4	IS_UART_FIFO_LEVEL	553
8.20.2.5	IS_UART_FLAG	553
8.20.2.6	IS_UART_GET_IT_SOURCE	554
8.20.2.7	IS_UART_PARITY_BIT	554
8.20.2.8	IS_UART_STOP_BIT	554
8.20.3	Перечисления	554
8.20.3.1	UART_DataWidth_TypeDef	554
8.20.3.2	UART_Dir_Typedef	555
8.20.3.3	UART_Error_Typedef	555
8.20.3.4	UART_FIFOLevel_TypeDef	555
8.20.3.5	UART_Flag_Typedef	556
8.20.3.6	UART_ITSource_Typedef	556
8.20.3.7	UART_ParityBit_TypeDef	556
8.20.3.8	UART_StopBit_TypeDef	557
8.20.4	Функции	557
8.20.4.1	UART_BaudRateDivConfig(NT_UART_TypeDef *UARTx, uint32_↵ t IntDiv, uint32_t FracDiv)	557
8.20.4.2	UART_Break(NT_UART_TypeDef *UARTx, FunctionalState State) .	557
8.20.4.3	UART_Cmd(NT_UART_TypeDef *UARTx, FunctionalState State) . .	557
8.20.4.4	UART_DeInit(NT_UART_TypeDef *UARTx)	558

8.20.4.5	UART_DMABlkOnErrCmd(NT_UART_TypeDef *UARTx, FunctionalState State)	558
8.20.4.6	UART_DMACmd(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)	558
8.20.4.7	UART_ErrorStatus(NT_UART_TypeDef *UARTx, UART_Error_TypeDef UART_Error)	559
8.20.4.8	UART_ErrorStatusClear(NT_UART_TypeDef *UARTx)	559
8.20.4.9	UART_FlagStatus(NT_UART_TypeDef *UARTx, UART_Flag_TypeDef UART_Flag)	559
8.20.4.10	UART_Init(NT_UART_TypeDef *UARTx, UART_Init_TypeDef *UART_InitStruct)	559
8.20.4.11	UART_ITCmd(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)	560
8.20.4.12	UART_ITFIFOLevelConfig(NT_UART_TypeDef *UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)	560
8.20.4.13	UART_ITMaskedStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	561
8.20.4.14	UART_ITRawStatus(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	562
8.20.4.15	UART_ITStatusClear(NT_UART_TypeDef *UARTx, UART_ITSource_TypeDef UART_ITSource)	562
8.20.4.16	UART_ModemConfig(NT_UART_TypeDef *UARTx, UART_ModemInit_TypeDef *UART_ModemInitStruct)	562
8.20.4.17	UART_ModemStructInit(UART_ModemInit_TypeDef *UART_ModemInitStruct)	563
8.20.4.18	UART_RecieveData(NT_UART_TypeDef *UARTx)	563
8.20.4.19	UART_SendData(NT_UART_TypeDef *UARTx, uint32_t Data)	563
8.20.4.20	UART_StructInit(UART_Init_TypeDef *UART_InitStruct)	564
8.21	Файл niietcm4_userflash.c	564
8.21.1	Подробное описание	565
8.21.2	Функции	565
8.21.2.1	USERFLASH_FullErase()	565
8.21.2.2	USERFLASH_Info_PageErase(uint32_t PageNum)	566
8.21.2.3	USERFLASH_Info_Read(uint32_t Address)	566
8.21.2.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	566
8.21.2.5	USERFLASH_Init(uint32_t SysClkFreq)	566
8.21.2.6	USERFLASH_ITCmd(FunctionalState State)	567
8.21.2.7	USERFLASH_OperationStatus()	567
8.21.2.8	USERFLASH_OperationStatusClear()	567
8.21.2.9	USERFLASH_PageErase(uint32_t PageNum)	567
8.21.2.10	USERFLASH_Read(uint32_t Address)	568
8.21.2.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	568
8.22	Файл niietcm4_userflash.h	568

8.22.1	Подробное описание	570
8.22.2	Макросы	570
8.22.2.1	IS_USERFLASH_STATUS	570
8.22.2.2	USERFLASH_INFO_PAGE_SIZE_BYTES	571
8.22.2.3	USERFLASH_INFO_PAGE_TOTAL	571
8.22.2.4	USERFLASH_INFO_TOTAL_BYTES	571
8.22.2.5	USERFLASH_PAGE_SIZE_BYTES	571
8.22.2.6	USERFLASH_PAGE_TOTAL	571
8.22.2.7	USERFLASH_TOTAL_BYTES	571
8.22.3	Перечисления	571
8.22.3.1	USERFLASH_Status_TypeDef	571
8.22.4	Функции	572
8.22.4.1	USERFLASH_FullErase()	572
8.22.4.2	USERFLASH_Info_PageErase(uint32_t PageNum)	572
8.22.4.3	USERFLASH_Info_Read(uint32_t Address)	572
8.22.4.4	USERFLASH_Info_Write(uint32_t Address, uint32_t Data)	572
8.22.4.5	USERFLASH_Init(uint32_t SysClkFreq)	573
8.22.4.6	USERFLASH_ITCmd(FunctionalState State)	573
8.22.4.7	USERFLASH_OperationStatus()	573
8.22.4.8	USERFLASH_OperationStatusClear()	573
8.22.4.9	USERFLASH_PageErase(uint32_t PageNum)	574
8.22.4.10	USERFLASH_Read(uint32_t Address)	574
8.22.4.11	USERFLASH_Write(uint32_t Address, uint32_t Data)	574
	Алфавитный указатель	575

Глава 1

Титульная страница

NIETCM4 PD (Peripheral driver) - это комплект драйверов, предназначенных для ускорения и упрощения работы со внутренними периферийными устройствами микроконтроллеров на базе ядра ARM Cortex-M4 производства ОАО "НИИЭТ". Совместно с комплектом предоставляются:

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

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

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

Загрузки

Текущую версию драйвера можно скачать по данной ссылке: [NIETCM4 Peripheral Driver v0.7.0](#).
Предыдущие версии могут быть найдены на странице [Загрузки](#).

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@niet.ru;

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

Глава 2

Информация о релизах

v0.7.0

Добавлено:

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

Исключено:

Исправлено:

- мелкие неточности документации функций блока UART, DMA, RCC

v0.6.0

Добавлено:

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

Исключено:

Исправлено:

- мелкие неточности документации блока UART

v0.5.0

Добавлено:

- драйвер для управления блоками таймеров
- драйвер для RTC

Исключено:

Исправлено:

v0.4.0

Добавлено:

- драйвер для управления UART

Исключено:

Исправлено:

- баг невозможности инициализации нескольких пинов одного порта отдельно

v0.3.0

Добавлено:

- драйвер для управления DMA

Исключено:

Исправлено:

v0.2.0

Добавлено:

- драйвер для управления тактированием и сбросом (RCC) K1921BK01T

Исключено:

Исправлено:

- структура документации GPIO
- форматирование и структура драйвера GPIO
- некритичные изменения в общей архитектуре

v0.1.0

Добавлено:

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

Исключено:

Исправлено:

Глава 3

Алфавитный указатель групп

3.1 Группы

Полный список групп.

Драйвер периферии	320
Настройка драйвера	321
Макросы	322
Типы	323
Периферия	325
ADC	210
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	50
Конфигурация прерываний	52
Цифровые компараторы	53
Секвенсоры	58
Приватные данные	212
Приватные константы	213
Начальные значения регистров	214
Приватные функции	215
BOOTFLASH	233
Константы	61
Основная область флеш	62
Информационная область флеш	63
Типы	64
Функции	65
Основная область флеш	67
Информационная область флеш	70
Приватные данные	234
Приватные константы	235
Приватные функции	236
DMA	240
Константы	72

Маски для CHANNEL_CFG	73
Маски каналов DMA	76
Маски каналов по имени	78
Маски каналов по номеру	82
Типы	86
Функции	91
Инициализация каналов DMA	92
Инициализация контроллера DMA	94
Конфигурация контроллера DMA	96
Статусная информация	100
Приватные данные	241
Приватные константы	242
Начальные значения регистров	243
Приватные функции	244
EXTMEM	254
Константы	103
Маски адреса	104
Типы	106
Функции	110
Приватные данные	255
Приватные константы	256
Начальные значения регистров	257
Приватные функции	258
GPIO	261
Типы	113
Константы	120
Маски пинов	121
Функции	125
Инициализация и деинициализация	126
Чтение и запись	128
Чтение	129
Запись	131
Битовые операции	133
Фильтрация	135
Прерывания	137
Приватные данные	263
Приватные константы	264
Начальные значения регистров	265
Маски портов	268
Приватные функции	270
RCC	278
Константы	139
Типы	140
Функции	149
Конфигурация PLL	151
Управление тактированием	154
Тактирование USB	156
Тактирование UART	158
Тактирование SPI	160
Тактирование ADC	162
Управление сбросом	164
Приватные данные	279
Приватные константы	280
Начальные значения регистров	281
Приватные функции	282
RTC	291

Типы	165
Функции	168
Приватные данные	292
Приватные функции	293
TIMER	294
Типы	169
Константы	171
Функции	172
Конфигурация	173
Прерывания	177
Приватные данные	295
Приватные константы	296
Приватные функции	297
UART	302
Типы	179
Константы	186
Функции	187
Инициализация и деинициализация	189
Прием и передача	191
Режим модема	193
Прерывания	195
Настройка DMA	198
Приватные данные	303
Приватные константы	304
Приватные функции	305
USERFLASH	313
Константы	200
Основная область флеш	201
Информационная область флеш	202
Типы	203
Функции	204
Основная область флеш	206
Информационная область флеш	208
Приватные данные	314
Приватные константы	315
Приватные функции	316

Глава 4

Алфавитный указатель структур данных

4.1 Структуры данных

Структуры данных с их кратким описанием.

_CHANNEL_CFG_bits	
Битовый доступ к регистру CHANNEL_CFG в DMA_Channel_TypeDef	327
ADC_DC_Init_TypeDef	
Структура инициализации цифровых компараторов	329
ADC_Init_TypeDef	
Структура инициализации модулей АЦП	331
ADC_SEQ_Init_TypeDef	
Структура инициализации секвенсоров	332
DMA_Channel_TypeDef	
Тип, описывающий структуру канала DMA	333
DMA_ChannelInit_TypeDef	
Структура инициализации канала DMA	335
DMA_ConfigData_TypeDef	
Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ	337
DMA_ConfigStruct_TypeDef	
Управляющая структура данных DMA	339
DMA_Init_TypeDef	
Структура инициализации контроллера DMA	340
DMA_Protect_TypeDef	
Защита шины при чтении из источника или записи в приемник через DMA	342
EXTMEM_Init_TypeDef	
Структура инициализации внешней памяти	343
GPIO_Init_TypeDef	
Структура инициализации GPIO	344
RCC_PLLInit_TypeDef	
Структура инициализации PLL	345
RTC_Date_TypeDef	
Структура даты	347
RTC_Time_TypeDef	
Структура времени	348
UART_Init_TypeDef	
Структура инициализации UART	349
UART_ModemInit_TypeDef	
Структура инициализации модемного режима	351

Глава 5

Список файлов

5.1 Файлы

Полный список документированных файлов.

niietcm4.h	Это главный заголовочный файл драйвера, обычно включаемый в main.c	353
niietcm4_adc.c	Файл содержит реализацию всех функции для работы с модулями АЦП, секвенсорами, цифровыми компараторами	356
niietcm4_adc.h	Файл содержит все прототипы функций для работы с АЦП, секвенсорами, цифровыми компараторами	374
niietcm4_bootflash.c	Файл содержит реализацию всех функции для работы с загрузочной флеш	411
niietcm4_bootflash.h	Файл содержит все прототипы функций для загрузочной флеш	415
niietcm4_conf.h	Файл конфигурации драйвера	422
niietcm4_dma.c	Файл содержит реализацию всех функции для работы с DMA	423
niietcm4_dma.h	Файл содержит все прототипы функций для DMA	431
niietcm4_extmem.c	Файл содержит реализацию всех функции для работы с интерфейсом внешней памяти	453
niietcm4_extmem.h	Файл содержит все прототипы функций для интерфейса внешней памяти	455
niietcm4_gpio.c	Файл содержит реализацию всех функции для работы с модулями GPIO	463
niietcm4_gpio.h	Файл содержит все прототипы функций для GPIO	475
niietcm4_rcc.c	Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера	493
niietcm4_rcc.h	Файл содержит все прототипы функций для RCC (Reset & Clock Control)	504
niietcm4_rtc.c	Файл содержит реализацию всех функции для работы с RTC	523
niietcm4_rtc.h	Файл содержит все прототипы функций для таймеров	524
niietcm4_timer.c	Файл содержит реализацию всех функции для работы с таймерами	528

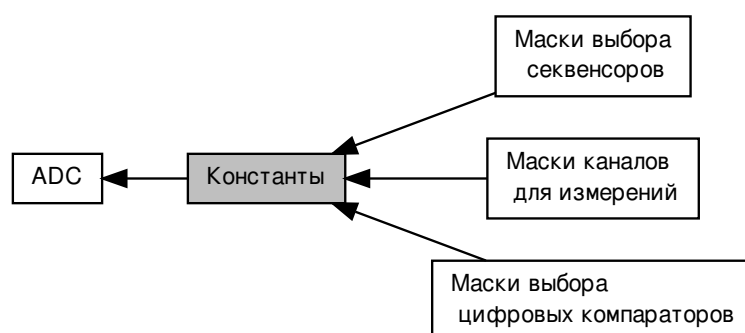
niietcm4_timer.h	
Файл содержит все прототипы функций для таймеров	533
niietcm4_uart.c	
Файл содержит реализацию всех функции для работы с модулями UART	538
niietcm4_uart.h	
Файл содержит все прототипы функций для UART	549
niietcm4_userflash.c	
Файл содержит реализацию всех функции для работы с пользовательской флеш .	564
niietcm4_userflash.h	
Файл содержит все прототипы функций для пользовательской флеш	568

Глава 6

Группы

6.1 Константы

Граф связей класса Константы:



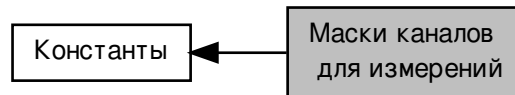
Группы

- [Маски каналов для измерений](#)
- [Маски выбора цифровых компараторов](#)
- [Маски выбора секвенсоров](#)

6.1.1 Подробное описание

6.2 Маски каналов для измерений

Граф связей класса 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)0x0FFFFFFF)
• #define IS_ADC_CHANNEL(CHANNEL) (((CHANNEL) & (uint32_t)0xFF000000) ==
((uint32_t)0x000000))
  
```

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)`

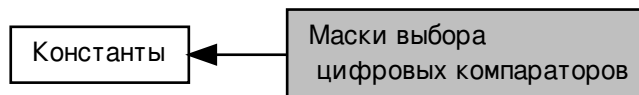
Канал ADC не выбран

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

Используется в `ADC_SEQ_StructInit()`.

6.3 Маски выбора цифровых компараторов

Граф связей класса 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)`

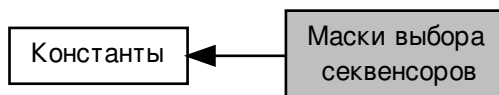
Цифровой компаратор не выбран

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

Используется в `ADC_SEQ_StructInit()`.

6.4 Маски выбора секвенсоров

Граф связей класса 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\leftarrowFFFFFF00) == ((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

Граф связей класса 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 вызывается 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) || \
  ((MODULE) == ADC_Module_12) || \
  ((MODULE) == ADC_Module_13) || \
  ((MODULE) == ADC_Module_14) || \
  ((MODULE) == ADC_Module_15) || \
  ((MODULE) == ADC_Module_16) || \
  ((MODULE) == ADC_Module_17) || \
  ((MODULE) == ADC_Module_18) || \
  ((MODULE) == ADC_Module_19) || \
  ((MODULE) == ADC_Module_20) || \
  ((MODULE) == ADC_Module_21) || \
  ((MODULE) == ADC_Module_22) || \
  ((MODULE) == ADC_Module_23))

```

```
((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_DMACmd()`, `ADC_SEQ_DMAConfig()`, `ADC_SEQ_DMAErrorStatus()`, `ADC_SEQ_DMAErrorStatusClear()`, `ADC_SEQ_FIFOEmptyStatus()`, `ADC_SEQ_FIFOEmptyStatusClear()`, `ADC_SEQ_FIFOFullStatus()`, `ADC_SEQ_FIFOFullStatusClear()`.

Q_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) || \
  ADC_SEQ_StartEvent_CMP0) || \
  ADC_SEQ_StartEvent_CMP1) || \
  ADC_SEQ_StartEvent_CMP2) || \
  ADC_SEQ_StartEvent_ITGPIO) || \
  ADC_SEQ_StartEvent_TIM) || \
  ADC_SEQ_StartEvent_PWM0) || \
  ADC_SEQ_StartEvent_PWM1) || \
  ADC_SEQ_StartEvent_PWM2) || \
  ADC_SEQ_StartEvent_PWM3) || \
  ADC_SEQ_StartEvent_PWM4) || \
  ADC_SEQ_StartEvent_PWM5) || \
  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.

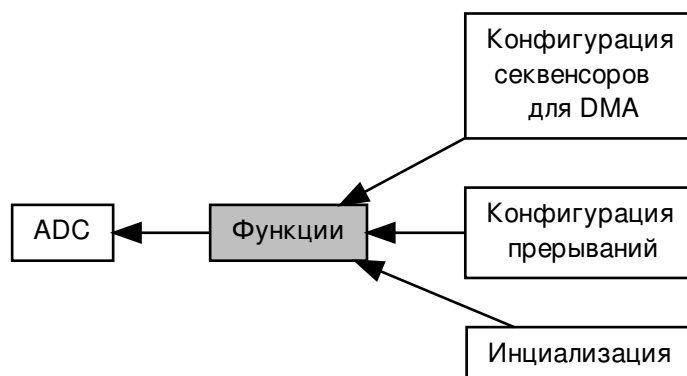
Элементы перечислений

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

Граф связей класса Функции:



Группы

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

Функции

- void [ADC_Cmd](#) (ADC_Module_TypeDef ADC_Module, [FunctionalState](#) State)
Включение модуля АЦП.
- void [ADC_SEQ_Cmd](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, [FunctionalState](#) State)
Включение секвенсора.
- void [ADC_SEQ_SWReq](#) ()
Программный запуск измерений всех разрешенных секвенсоров.
- uint32_t [ADC_SEQ_GetFIFOData](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Получение результата измерений из буфера секвенсора.
- uint32_t [ADC_SEQ_GetConversionCount](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.
- uint32_t [ADC_SEQ_GetFIFOLoad](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Получение количества измерений, сохраненных в буфере секвенсора.
- [FlagStatus](#) [ADC_SEQ_FIFOFullStatus](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.
- void [ADC_SEQ_FIFOFullStatusClear](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Сброс флага заполнения буфера секвенсора.
- [FlagStatus](#) [ADC_SEQ_FIFOEmptyStatus](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.
- void [ADC_SEQ_FIFOEmptyStatusClear](#) (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

- Сброс флага пустоты буфера секвенсора.
- void [ADC_DC_Cmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
Включение выходного триггера цифрового компаратора.
- [FlagStatus](#) [ADC_DC_TrigStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Проверка состояния выходного триггера компаратора.
- void [ADC_DC_TrigStatusClear](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Сброс выходного триггера цифрового компаратора.
- uint32_t [ADC_DC_GetLastData](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.

6.6.1 Подробное описание

6.6.2 Функции

6.6.2.1 void ADC_Cmd (ADC_Module_TypeDef ADC_Module, FunctionalState State)

Включение модуля АЦП.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 115

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

6.6.2.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1024

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

6.6.2.3 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1040

Перекрестные ссылки IS_ADC_DC_MODULE.

6.6.2.4 FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1058

Перекрестные ссылки IS_ADC_DC_MODULE.

6.6.2.5 void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Сброс выходного триггера цифрового компаратора.

Внимание

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

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1086

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

6.6.2.6 void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 855

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

6.6.2.7 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 983

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.8 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1008

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.9 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 943

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.10 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 968

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.11 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 905

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.12 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение результата измерений из буфера секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 887

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.13 uint32_t ADC_SEQ_GetFIFOLoad (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 923

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.6.2.14 void ADC_SEQ_SWReq ()

Программный запуск измерений всех разрешенных секвенсоров.

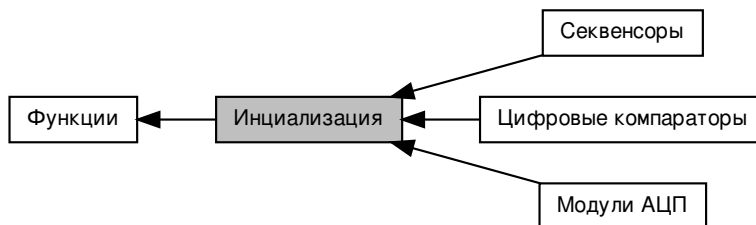
Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 875

6.7 Инициализация

Граф связей класса Инициализация:



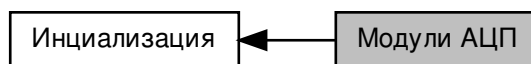
Группы

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

6.7.1 Подробное описание

6.8 Модули АЦП

Граф связей класса Модули АЦП:



Функции

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

6.8.1 Подробное описание

6.8.2 Функции

6.8.2.1 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 130

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

6.8.2.2 void ADC_Init (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *ADC_InitStruct)

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

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

См. определение в файле niietcm4_adc.c строка 206

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_InitStruct	Указатель на структуру типа ADC_Init_TypeDef , которую необходимо проинициализировать.
----------------	--

Возвращаемые значения

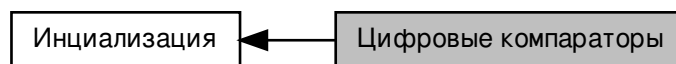
Нет

См. определение в файле niietcm4_adc.c строка 290

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

Граф связей класса 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

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

Возвращаемые значения

Нет

См. определение в файле [niietcm4_adc.c](#) строка 307

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

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

См. определение в файле niietcm4_adc.c строка 326

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_THRESHO↔LD](#).

6.9.2.3 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

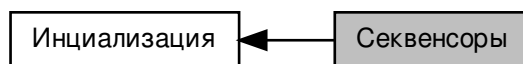
Нет

См. определение в файле niietcm4_adc.c строка 349

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

Граф связей класса Секвенсоры:



Функции

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

6.10.1 Подробное описание

6.10.2 Функции

6.10.2.1 void [ADC_SEQ_DeInit](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 365

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 ADC_SEQ_↔ Module_TypeDef .
ADC_SEQ_↔ InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 425

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 ADC_SEQ_Init_TypeDef , которую необходимо проинициализировать.
-------------------------	--

Возвращаемые значения

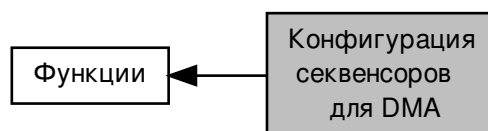
Нет

См. определение в файле niietcm4_adc.c строка 460

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::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

Граф связей класса Конфигурация секвенсоров для DMA:



Функции

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

6.11.1 Подробное описание

6.11.2 Функции

6.11.2.1 `void ADC_SEQ_DMACmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)`

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_adc.c` строка 496

Перекрестные ссылки `IS_ADC_SEQ_MODULE` и `IS_FUNCTIONAL_STATE`.

6.11.2.2 `void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)`

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 479

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

6.11.2.3 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 512

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.11.2.4 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

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

Возвращаемые значения

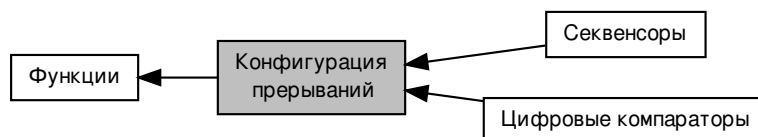
нет

См. определение в файле niietcm4_adc.c строка 537

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.12 Конфигурация прерываний

Граф связей класса Конфигурация прерываний:



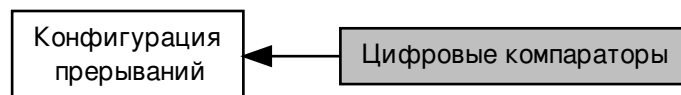
Группы

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

6.12.1 Подробное описание

6.13 Цифровые компараторы

Граф связей класса Цифровые компараторы:



Функции

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

6.13.1 Подробное описание

6.13.2 Функции

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

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 596

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

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

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 620

Перекрестные ссылки [IS_ADC_DC_CONDITION](#), [IS_ADC_DC_MODE](#) и [IS_ADC_DC_MODULE](#).

```
6.13.2.3 void ADC_DC_ITGenCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                               FunctionalState State )
```

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 553

Перекрестные ссылки [IS_ADC_DC_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

Используется в [ADC_DC_ITCmd\(\)](#).

```
6.13.2.4 void ADC_DC_ITMaskCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,  
    FunctionalState State )
```

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 570

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 ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 662

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 ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 637

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 ADC_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

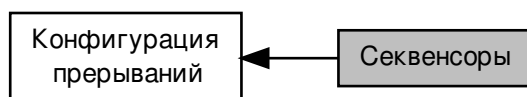
нет	
-----	--

См. определение в файле niietcm4_adc.c строка 688

Перекрестные ссылки IS_ADC_DC_MODULE.

6.14 Секвенсоры

Граф связей класса Секвенсоры:



Функции

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

6.14.1 Подробное описание

6.14.2 Функции

6.14.2.1 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 757

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.2 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 704

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

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

Настройка вызова прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 732

Перекрестные ссылки IS_ADC_SEQ_IT_RATE, IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

6.14.2.4 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 775

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.5 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 814

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.6 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 789

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.14.2.7 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

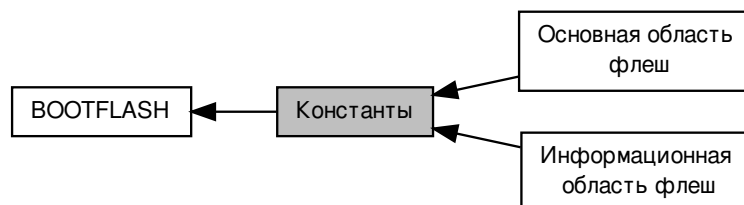
нет	
-----	--

См. определение в файле niietcm4_adc.c строка 839

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.15 Константы

Граф связей класса Константы:



Группы

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

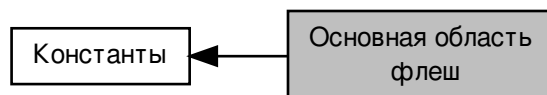
Макросы

- `#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

Граф связей класса 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

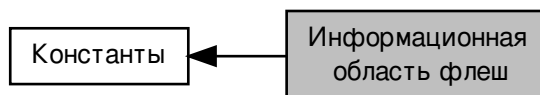
6.16.2.3 `#define BOOTFLASH_TOTAL_BYTES (BOOTFLASH_PAGE_SIZE_BYTES*BOOTFLASH_PAGE_TOTAL)`

Общий размер основной области.

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

6.17 Информационная область флеш

Граф связей класса 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

Граф связей класса 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.

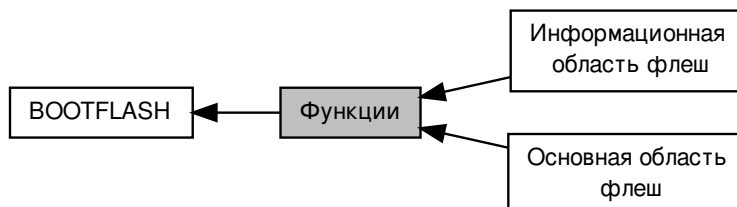
`BOOTFLASH_Status_Complete` Операция успешно завершена.

`BOOTFLASH_Status_Error` Операция завершена с ошибкой.

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

6.19 Функции

Граф связей класса 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

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 75

6.19.2.2 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.19.2.3 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_bootflash.c строка 88

6.19.2.4 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

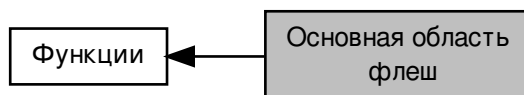
Возвращаемые значения

Нет.

См. определение в файле niietcm4_bootflash.c строка 102

6.20 Основная область флеш

Граф связей класса 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

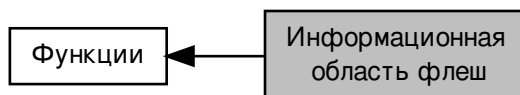
Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 128

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.21 Информационная область флеш

Граф связей класса 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

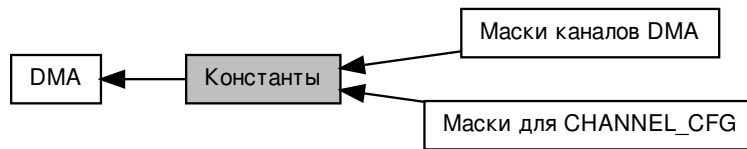
Нет

См. определение в файле niietcm4_bootflash.c строка 163

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.22 Константы

Граф связей класса Константы:



Группы

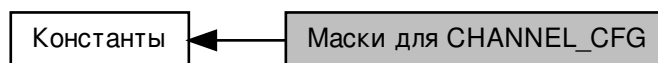
- [Маски для CHANNEL_CFG](#)
Битовые позиции и маски регистра CHANNEL_CFG в [DMA_Channel_TypeDef](#).
- [Маски каналов DMA](#)

6.22.1 Подробное описание

6.23 Маски для CHANNEL_CFG

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

Граф связей класса Маски для CHANNEL_CFG:



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)`

6.23.1 Подробное описание

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

6.23.2 Макросы

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

Поле задания типа цикла DMA

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

6.23.2.2 `#define CHANNEL_CFG_CYCLE_CTRL_Pos 0`

Поле задания типа цикла DMA

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

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

Шаг инкремента адреса приемника

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

6.23.2.4 `#define CHANNEL_CFG_DST_INC_Pos 30`

Шаг инкремента адреса приемника

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

6.23.2.5 `#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

6.23.2.6 `#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

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

Разрядность данных приемника

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

6.23.2.8 `#define CHANNEL_CFG_DST_SIZE_Pos 28`

Разрядность данных приемника

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

6.23.2.9 `#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

6.23.2.10 `#define CHANNEL_CFG_N_MINUS_1_Pos 4`

Общее количество передач в цикле работы

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

6.23.2.11 `#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

6.23.2.12 `#define CHANNEL_CFG_NEXT_USEBURST_Pos 3`

Контролирует установку соответствующего канала бита в регистре NT_DMA->CHNL_USEBURST_SET

См. определение в файле niietcm4_dma.h строка 58

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

Количество передач до выполнения переарбитрации

См. определение в файле niietcm4_dma.h строка 71

6.23.2.14 `#define CHANNEL_CFG_R_POWER_Pos 14`

Количество передач до выполнения переарбитрации

См. определение в файле niietcm4_dma.h строка 60

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

Шаг инкремента адреса источника

См. определение в файле niietcm4_dma.h строка 75

6.23.2.16 `#define CHANNEL_CFG_SRC_INC_Pos 26`

Шаг инкремента адреса источника

См. определение в файле niietcm4_dma.h строка 64

6.23.2.17 `#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

6.23.2.18 `#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

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

Разрядность данных источника

См. определение в файле niietcm4_dma.h строка 74

6.23.2.20 `#define CHANNEL_CFG_SRC_SIZE_Pos 24`

Разрядность данных источника

См. определение в файле niietcm4_dma.h строка 63

6.24 Маски каналов DMA

Граф связей класса Маски каналов DMA:



Группы

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

Макросы

- `#define DMA_Channel_All ((uint32_t)0x00FFFFFF)`
- `#define IS_DMA_CHANNEL(CHANNEL) (((CHANNEL) != (uint32_t)0x000000) && (((CHANNEL) & (uint32_t)0xFF000000) == ((uint32_t)0x0000)))`
Макрос проверки маски каналов на попадание в допустимый диапазон.
- `#define IS_GET_DMA_CHANNEL(CHANNEL)`
Макрос проверки маски канала при работе с каналами по отдельности.

6.24.1 Подробное описание

6.24.2 Макросы

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

Все каналы DMA

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

6.24.2.2 `#define IS_GET_DMA_CHANNEL(CHANNEL)`

Макроопределение:

```

(((CHANNEL) == (DMA_Channel_0 || DMA_Channel_UART0_TX)) || \
 DMA_Channel_UART1_TX) || \
 ((CHANNEL) == (DMA_Channel_2 || \
 DMA_Channel_UART2_TX)) || \
 ((CHANNEL) == (DMA_Channel_3 || \
 DMA_Channel_UART3_TX)) || \
 ((CHANNEL) == (DMA_Channel_4 || \
 DMA_Channel_UART0_RX)) || \
 ((CHANNEL) == (DMA_Channel_5 || \
 DMA_Channel_UART1_RX)) || \
 ((CHANNEL) == (DMA_Channel_6 || \
 DMA_Channel_UART2_RX)) || \
 ((CHANNEL) == (DMA_Channel_7 ||

```

```

DMA_Channel_UART3_RX)) || \
    ((CHANNEL) == (DMA_Channel_8 ||
DMA_Channel_ADCSEQ0)) || \
    ((CHANNEL) == (DMA_Channel_9 ||
DMA_Channel_ADCSEQ1)) || \
    ((CHANNEL) == (DMA_Channel_10 ||
DMA_Channel_ADCSEQ2)) || \
    ((CHANNEL) == (DMA_Channel_11 ||
DMA_Channel_ADCSEQ3)) || \
    ((CHANNEL) == (DMA_Channel_12 ||
DMA_Channel_ADCSEQ4)) || \
    ((CHANNEL) == (DMA_Channel_13 ||
DMA_Channel_ADCSEQ5)) || \
    ((CHANNEL) == (DMA_Channel_14 ||
DMA_Channel_ADCSEQ6)) || \
    ((CHANNEL) == (DMA_Channel_15 ||
DMA_Channel_ADCSEQ7)) || \
    ((CHANNEL) == (DMA_Channel_16 ||
DMA_Channel_SPI0_TX)) || \
    ((CHANNEL) == (DMA_Channel_17 ||
DMA_Channel_SPI1_TX)) || \
    ((CHANNEL) == (DMA_Channel_18 ||
DMA_Channel_SPI2_TX)) || \
    ((CHANNEL) == (DMA_Channel_10 ||
DMA_Channel_SPI3_TX)) || \
    ((CHANNEL) == (DMA_Channel_20 ||
DMA_Channel_SPI0_RX)) || \
    ((CHANNEL) == (DMA_Channel_21 ||
DMA_Channel_SPI1_RX)) || \
    ((CHANNEL) == (DMA_Channel_22 ||
DMA_Channel_SPI2_RX)) || \
    ((CHANNEL) == (DMA_Channel_23 ||
DMA_Channel_SPI3_RX)))

```

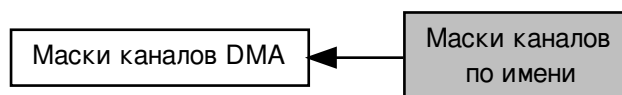
Макрос проверки маски канала при работе с каналами по отдельности.

См. определение в файле niietcm4_dma.h строка 166

Используется в DMA_WaitOnReqStatus().

6.25 Маски каналов по имени

Граф связей класса Маски каналов по имени:



Макросы

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

6.25.1 Подробное описание

6.25.2 Макросы

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

Канал DMA секвенсора 0 АЦП

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

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

Канал DMA секвенсора 1 АЦП

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

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

Канал DMA секвенсора 2 АЦП

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

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

Канал DMA секвенсора 3 АЦП

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

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

Канал DMA секвенсора 4 АЦП

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

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

Канал DMA секвенсора 5 АЦП

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

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

Канал DMA секвенсора 6 АЦП

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

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

Канал DMA секвенсора 7 АЦП

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

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

Канал DMA по приему от SPI0

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

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

Канал DMA по передаче от SPI0

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

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

Канал DMA по приему от SPI1

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

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

Канал DMA по передаче от SPI1

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

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

Канал DMA по приему от SPI2

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

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

Канал DMA по передаче от SPI2

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

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

Канал DMA по приему от SPI3

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

6.25.2.16 `#define DMA_Channel_SPI3_TX ((uint32_t)0x00080000)`

Канал DMA по передаче от SPI3

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

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

Канал DMA по приему от UART0

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

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

Канал DMA по передаче от UART0

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

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

Канал DMA по приему от UART1

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

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

Канал DMA по передаче от UART1

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

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

Канал DMA по приему от UART2

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

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

Канал DMA по передаче от UART2

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

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

Канал DMA по приему от UART3

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

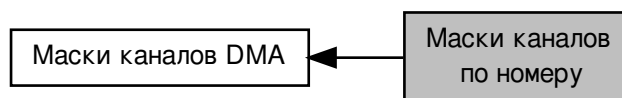
6.25.2.24 `#define DMA_Channel_UART3_TX ((uint32_t)0x00000008)`

Канал DMA по передаче от UART3

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

6.26 Маски каналов по номеру

Граф связей класса 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)`

6.26.1 Подробное описание

6.26.2 Макросы

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

Канал DMA 0

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

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

Канал DMA 1

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

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

Канал DMA 10

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

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

Канал DMA 11

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

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

Канал DMA 12

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

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

Канал DMA 13

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

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

Канал DMA 14

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

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

Канал DMA 15

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

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

Канал DMA 16

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

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

Канал DMA 17

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

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

Канал DMA 18

См. определение в файле niietcm4_dma.h строка 144

6.26.2.12 #define DMA_Channel_19 ((uint32_t)0x00080000)

Канал DMA 19

См. определение в файле niietcm4_dma.h строка 145

6.26.2.13 #define DMA_Channel_2 ((uint32_t)0x00000004)

Канал DMA 2

См. определение в файле niietcm4_dma.h строка 128

6.26.2.14 #define DMA_Channel_20 ((uint32_t)0x00100000)

Канал DMA 20

См. определение в файле niietcm4_dma.h строка 146

6.26.2.15 #define DMA_Channel_21 ((uint32_t)0x00200000)

Канал DMA 21

См. определение в файле niietcm4_dma.h строка 147

6.26.2.16 #define DMA_Channel_22 ((uint32_t)0x00400000)

Канал DMA 22

См. определение в файле niietcm4_dma.h строка 148

6.26.2.17 #define DMA_Channel_23 ((uint32_t)0x00800000)

Канал DMA 23

См. определение в файле niietcm4_dma.h строка 149

6.26.2.18 #define DMA_Channel_3 ((uint32_t)0x00000008)

Канал DMA 3

См. определение в файле niietcm4_dma.h строка 129

6.26.2.19 #define DMA_Channel_4 ((uint32_t)0x00000010)

Канал DMA 4

См. определение в файле niietcm4_dma.h строка 130

6.26.2.20 #define DMA_Channel_5 ((uint32_t)0x00000020)

Канал DMA 5

См. определение в файле niietcm4_dma.h строка 131

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

Канал DMA 6

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

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

Канал DMA 7

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

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

Канал DMA 8

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

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

Канал DMA 9

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

6.27 Типы

Граф связей класса 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_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_Pause`,
`DMA_State_Done`, `DMA_State_PeriphScatGath` }

Возможные состояния конечного автомата управления контроллером DMA.

6.27.1 Подробное описание

6.27.2 Макросы

6.27.2.1 `#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()`.

6.27.2.2 `#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()`.

6.27.2.3 `#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()`.

6.27.2.4 `#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()`.

6.27.2.5 `#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 458

6.27.3 Перечисления

6.27.3.1 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

6.27.3.2 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

6.27.3.3 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

6.27.3.4 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

6.27.3.5 enum DMA_State_TypeDef

Возможные состояния конечного автомата управления контроллером DMA.

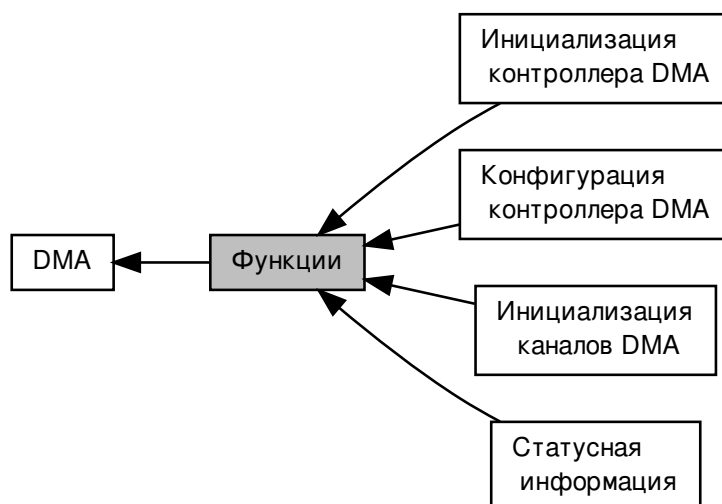
Элементы перечислений

- 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_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

6.28 Функции

Граф связей класса Функции:



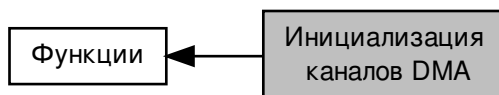
Группы

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

6.28.1 Подробное описание

6.29 Инициализация каналов DMA

Граф связей класса Инициализация каналов DMA:



Функции

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

6.29.1 Подробное описание

6.29.2 Функции

6.29.2.1 void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)

Деинициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 87

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

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

Инициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 102

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Channel_TypeDef::CHANNEL_CFG_bit, _CHANNEL_CFG_bits::CYCLE_CTRL, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_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.

6.29.2.3 void DMA_ChannelStructInit (DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

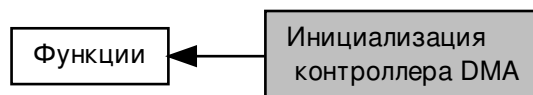
Нет

См. определение в файле niietcm4_dma.c строка 146

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

6.30 Инициализация контроллера DMA

Граф связей класса 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.

6.30.1 Подробное описание

6.30.2 Функции

6.30.2.1 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет

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

6.30.2.2 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

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

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 195

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

6.30.2.3 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

Аргументы

DMA_InitStruct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
----------------	--

Возвращаемые значения

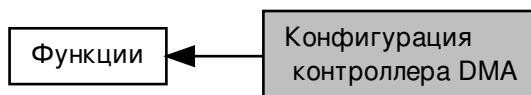
Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 212

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Init_TypeDef::DMA_Channel, DMA_Init_TypeDef::DMA_ChannelEnable, DMA_Init_TypeDef::DMA_HighPriority, DMA_Init_TypeDef::DMA_PrmAlt, DMA_Init_TypeDef::DMA_Protection, DMA_Init_TypeDef::DMA_ReqMask, DMA_Init_TypeDef::DMA_UseBurst и DMA_Protect_TypeDef::PRIVELGED.

6.31 Конфигурация контроллера DMA

Граф связей класса 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.

6.31.1 Подробное описание

6.31.2 Функции

6.31.2.1 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

6.31.2.2 void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)

Активация каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 373

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

6.31.2.3 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

6.31.2.4 void DMA_MasterEnableCmd (FunctionalState State)

Разрешения работы контроллера DMA.

Внимание

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

Аргументы

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

Возвращаемые значения

Нет	
-----	--

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

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

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

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.31.2.6 `void DMA_ProtectionConfig (DMA_Protect_TypeDef * DMA_Protection)`

Управление защитой шины при обращении DMA к управляющим данным.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

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

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

Используется в `DMA_Init()`.

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

Маскирование каналов DMA.

Внимание

По маскированным каналам игнорируются запросы на передачи.

Аргументы

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

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.31.2.8 `void DMA_SWRequestCmd (uint32_t DMA_Channel)`

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL`.

6.31.2.9 `void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)`

Установка пакетного обмена каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

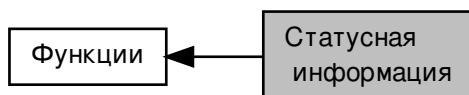
См. определение в файле `niietcm4_dma.c` строка 324

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.32 Статусная информация

Граф связей класса 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.

6.32.1 Подробное описание

6.32.2 Функции

6.32.2.1 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 524

6.32.2.2 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 503

6.32.2.3 FunctionalState DMA_MasterEnableStatus ()

Состояние контроллера DMA.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 455

6.32.2.4 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

6.32.2.5 FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

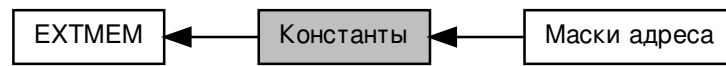
Status	Одно из значений FunctionalState: <ul style="list-style-type: none"> • ENABLE - поддерживаются SREQ (как и блочные BREQ); • DISABLE - поддерживаются только блочные запросы BREQ.
--------	---

См. определение в файле niietcm4_dma.c строка 478

Перекрестные ссылки IS_GET_DMA_CHANNEL.

6.33 Константы

Граф связей класса Константы:



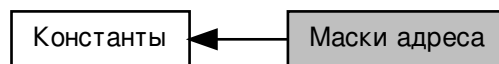
Группы

- [Маски адреса](#)

6.33.1 Подробное описание

6.34 Маски адреса

Граф связей класса 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.

6.34.1 Подробное описание

6.34.2 Макросы

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

Маска бита 11 адреса контроллера внешней памяти.

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

6.34.2.2 `#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

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

Маска бита 12 адреса контроллера внешней памяти.

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

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

Маска бита 13 адреса контроллера внешней памяти.

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

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

Маска бита 14 адреса контроллера внешней памяти.

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

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

Маска бита 15 адреса контроллера внешней памяти.

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

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

Маска бита 16 адреса контроллера внешней памяти.

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

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

Маска бита 17 адреса контроллера внешней памяти.

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

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

Маска бита 18 адреса контроллера внешней памяти.

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

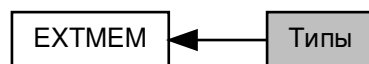
6.34.2.10 `#define EXTMEM_CEMask_Addr_19 ((uint32_t)0x0100)`

Маска бита 19 адреса контроллера внешней памяти.

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

6.35 Типы

Граф связей класса 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.

6.35.1 Подробное описание

6.35.2 Макросы

6.35.2.1 `#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()`.

6.35.2.2 `#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()`.

6.35.2.3 `#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()`.

6.35.2.4 `#define IS_EXTMEM_WRITE_WAITSTATE(WAITSTATE)`

Макроопределение:

```
(((WAITSTATE) == EXTMEM_WriteWaitState_1) || \
  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()`.

6.35.3 Перечисления

6.35.3.1 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

6.35.3.2 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

6.35.3.3 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

6.35.3.4 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

6.36 Функции

Граф связей класса 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.

6.36.1 Подробное описание

6.36.2 Функции

6.36.2.1 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](#).

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

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

Аргументы

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

Возвращаемые значения

Нет

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

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

6.36.2.3 void EXTMEM_StructInit (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Заполнение каждого члена структуры EXTMEM_InitStruct значениями по умолчанию.

Аргументы

EXTMEM_↔ InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 107

Перекрестные ссылки EXTMEM_Init_TypeDef::CEMask, EXTMEM_Init_TypeDef::EXTMEM_↔ReadWaitState, EXTMEM_ReadWaitState_8, EXTMEM_Init_TypeDef::EXTMEM_RWWaitState, EXTMEM_RWWaitState_1, EXTMEM_Init_TypeDef::EXTMEM_Width, EXTMEM_Width_↔16bit, EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState и EXTMEM_WriteWaitState_1.

6.37 Типы

Граф связей класса 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.

6.37.1 Подробное описание

6.37.2 Макросы

6.37.2.1 #define IS_GPIO_ALT_FUNC(ALT_FUNC)

Макроопределение:

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

Макрос проверки аргументов типа `GPIO_AltFunc_TypeDef`.

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

Используется в `GPIO_Init()`.

6.37.2.2 #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()`.

6.37.2.3 `#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()`.

6.37.2.4 `#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()`.

6.37.2.5 `#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()`.

6.37.2.6 `#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()`.

6.37.2.7 `#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()`.

6.37.2.8 `#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()`.

6.37.2.9 `#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()`.

6.37.2.10 `#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

6.37.2.11 `#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()`.

6.37.2.12 `#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

6.37.3 Перечисления

6.37.3.1 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

6.37.3.2 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

6.37.3.3 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

6.37.3.4 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

6.37.3.5 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

6.37.3.6 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

6.37.3.7 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

6.37.3.8 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

6.37.3.9 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

6.37.3.10 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

6.37.3.11 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

6.37.3.12 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

6.37.3.13 enum GPIO_Sync_TypeDef

Включение режима пересинхронизации входов через 2 триггера-защелки.

Элементы перечислений

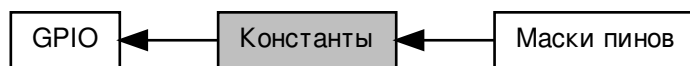
GPIO_Sync_Dis Пересинхронизация входа выключена.

GPIO_Sync_En Пересинхронизация входа включена.

См. определение в файле niietcm4_gpio.h строка 214

6.38 Константы

Граф связей класса Константы:



Группы

- [Маски пинов](#)

6.38.1 Подробное описание

6.39 Маски пинов

Граф связей класса 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) <= ((uint32_t)0x0000)))`

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.

6.39.1 Подробное описание

6.39.2 Макросы

6.39.2.1 `#define GPIO_Pin_0 ((uint32_t)0x0001)`

Пин 0 выбран.

См. определение в файле `niietcm4_gpio.h` строка 319

Используется в `RCC_SysClkDiv2Out()`.

6.39.2.2 `#define GPIO_Pin_0_3 ((uint32_t)0x000F)`

Пины 0-3 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 335

6.39.2.3 `#define GPIO_Pin_0_7 ((uint32_t)0x00FF)`

Пины 0-7 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 339

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

Пин 1 выбран.

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

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

Пин 10 выбран.

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

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

Пин 11 выбран.

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

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

Пин 12 выбран.

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

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

Пины 12-15 выбраны.

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

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

Пин 13 выбран.

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

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

Пин 14 выбран.

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

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

Пин 15 выбран.

См. определение в файле niietcm4_gpio.h строка 334

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

Пин 2 выбран.

См. определение в файле niietcm4_gpio.h строка 321

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

Пин 3 выбран.

См. определение в файле niietcm4_gpio.h строка 322

6.39.2.14 `#define GPIO_Pin_4 ((uint32_t)0x0010)`

Пин 4 выбран.

См. определение в файле niietcm4_gpio.h строка 323

6.39.2.15 `#define GPIO_Pin_4_7 ((uint32_t)0x00F0)`

Пины 4-7 выбраны.

См. определение в файле niietcm4_gpio.h строка 336

6.39.2.16 `#define GPIO_Pin_5 ((uint32_t)0x0020)`

Пин 5 выбран.

См. определение в файле niietcm4_gpio.h строка 324

6.39.2.17 `#define GPIO_Pin_6 ((uint32_t)0x0040)`

Пин 6 выбран.

См. определение в файле niietcm4_gpio.h строка 325

6.39.2.18 `#define GPIO_Pin_7 ((uint32_t)0x0080)`

Пин 7 выбран.

См. определение в файле niietcm4_gpio.h строка 326

6.39.2.19 `#define GPIO_Pin_8 ((uint32_t)0x0100)`

Пин 8 выбран.

См. определение в файле niietcm4_gpio.h строка 327

6.39.2.20 `#define GPIO_Pin_8_11 ((uint32_t)0x0F00)`

Пины 8-11 выбраны.

См. определение в файле niietcm4_gpio.h строка 337

6.39.2.21 `#define GPIO_Pin_8_15 ((uint32_t)0xFF00)`

Пины 8-15 выбраны.

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

6.39.2.22 `#define GPIO_Pin_9 ((uint32_t)0x0200)`

Пин 9 выбран.

См. определение в файле `niietcm4_gpio.h` строка 328

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

Все пины выбраны.

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

Используется в `GPIO_StructInit()`.

6.39.2.24 `#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))
```

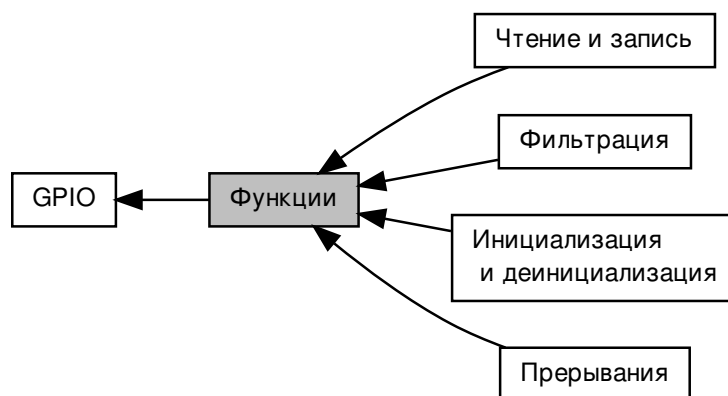
Макрос проверки номера пина при работе с пинами по отдельности.

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

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

6.40 Функции

Граф связей класса Функции:



Группы

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

6.40.1 Подробное описание

6.41 Инициализация и деинициализация

Граф связей класса Инициализация и деинициализация:



Функции

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

6.41.1 Подробное описание

6.41.2 Функции

6.41.2.1 void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)

Устанавливает все регистры выбранного GPIOx значениями по умолчанию.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Нет

См. определение в файле 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](#).

6.41.2.2 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

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

Перекрестные ссылки `GPIO_Init_TypeDef::GPIO_AltFunc`, `GPIO_Init_TypeDef::GPIO_Dir`, `GPIO_Dir_In`, `GPIO_Dir_Out`, `GPIO_Init_TypeDef::GPIO_Load`, `GPIO_Load_16mA`, `GPIO_Load_8mA`, `GPIO_Init_TypeDef::GPIO_Mode`, `GPIO_Mode_AltFunc`, `GPIO_Mode_IO`, `GPIO_Init_TypeDef::GPIO_Out`, `GPIO_Out_Dis`, `GPIO_Out_En`, `GPIO_Init_TypeDef::GPIO_OutMode`, `GPIO_OutMode_OD`, `GPIO_OutMode_PP`, `GPIO_Init_TypeDef::GPIO_Pin`, `GPIO_Init_TypeDef::GPIO_PullUp`, `GPIO_PullUp_Dis`, `GPIO_PullUp_En`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_ALT_FUNC`, `IS_GPIO_DIR`, `IS_GPIO_LOAD`, `IS_GPIO_MODE`, `IS_GPIO_OUT`, `IS_GPIO_OUT_MODE`, `IS_GPIO_PIN` и `IS_GPIO_PULLUP`.

Используется в `RCC_SysClkDiv2Out()`.

6.41.2.3 `void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)`

Заполнение каждого члена структуры `GPIO_InitStruct` значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет

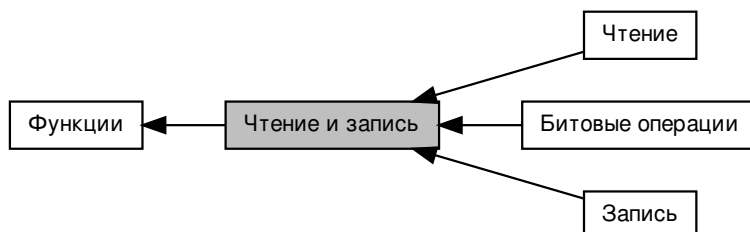
См. определение в файле `niietcm4_gpio.c` строка 456

Перекрестные ссылки `GPIO_Init_TypeDef::GPIO_AltFunc`, `GPIO_AltFunc_1`, `GPIO_Init_TypeDef::GPIO_Dir`, `GPIO_Dir_In`, `GPIO_Init_TypeDef::GPIO_Load`, `GPIO_Load_8mA`, `GPIO_Init_TypeDef::GPIO_Mode`, `GPIO_Mode_IO`, `GPIO_Init_TypeDef::GPIO_Out`, `GPIO_Out_Dis`, `GPIO_Init_TypeDef::GPIO_OutMode`, `GPIO_OutMode_PP`, `GPIO_Init_TypeDef::GPIO_Pin`, `GPIO_Pin_All`, `GPIO_Init_TypeDef::GPIO_PullUp` и `GPIO_PullUp_Dis`.

Используется в `RCC_SysClkDiv2Out()`.

6.42 Чтение и запись

Граф связей класса Чтение и запись:



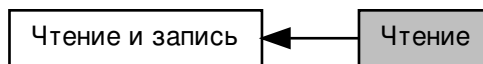
Группы

- [Чтение](#)
- [Запись](#)
- [Битовые операции](#)

6.42.1 Подробное описание

6.43 Чтение

Граф связей класса 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.

6.43.1 Подробное описание

6.43.2 Функции

6.43.2.1 uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx	
-----------------------	--

См. определение в файле niietcm4_gpio.c строка 503

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.43.2.2 uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

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

См. определение в файле `niietcm4_gpio.c` строка 477

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

6.43.2.3 `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 x лежит в диапазоне A..H.
<code>MaskVal</code>	Значение маски чтения.

Возвращаемые значения

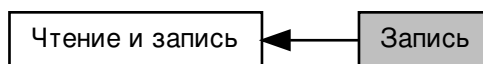
Состояние порта <code>GPIOx</code> с учетом маски	
--	--

См. определение в файле `niietcm4_gpio.c` строка 518

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

6.44 Запись

Граф связей класса 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.

6.44.1 Подробное описание

6.44.2 Функции

6.44.2.1 void [GPIO_Write](#) (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне А..Н.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки [IS_GPIO_ALL_PERIPH](#).

6.44.2.2 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 А..Н.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .
--------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 546

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

6.44.2.3 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 586

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.45 Битовые операции

Граф связей класса 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.

6.45.1 Подробное описание

6.45.2 Функции

6.45.2.1 void [GPIO_ClearBits](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 626

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.45.2.2 void [GPIO_SetBits](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 609

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.45.2.3 void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Переключение выбранных пинов в противоположное состояние.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

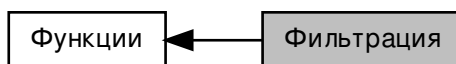
Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 643

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.46 Фильтрация

Граф связей класса Фильтрация:



Функции

- 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.

6.46.1 Подробное описание

6.46.2 Функции

6.46.2.1 void [GPIO_QualCmd](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.46.2.2 void [GPIO_QualConfig](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [GPIO_QualMode_TypeDef](#) Mode, uint32_t SamplePerod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_↵_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 664

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

```
6.46.2.3 void GPIO_SyncCmd ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,
FunctionalState State )
```

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

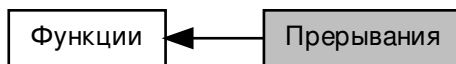
Нет

См. определение в файле niietcm4_gpio.c строка 814

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.47 Прерывания

Граф связей класса 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.

6.47.1 Подробное описание

6.47.2 Функции

6.47.2.1 void [GPIO_ITCmd](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niitcm4_gpio.c строка 913

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_GPIO_ALL_PERIPH](#) и [IS_GPIO_PIN](#).

6.47.2.2 void [GPIO_ITConfig](#) (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, [GPIO_IntType_TypeDef IntType](#), [GPIO_IntPol_TypeDef IntPol](#))

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 876

Перекрестные ссылки `GPIO_IntPol_Neg`, `GPIO_IntPol_Pos`, `GPIO_IntType_Edge`, `GPIO_IntType_Level`, `IS_GPIO_ALL_PERIPH`, `IS_GPIO_INT_POL`, `IS_GPIO_INT_TYPE` и `IS_GPIO_PIN`.

6.47.2.3 `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

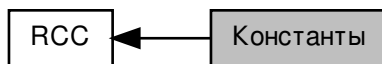
Нет

См. определение в файле `niietcm4_gpio.c` строка 938

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

6.48 Константы

Граф связей класса Константы:



Макросы

- `#define RCC_CLK_CHANGE_TIMEOUT ((uint32_t)10000)`

6.48.1 Подробное описание

6.48.2 Макросы

6.48.2.1 `#define RCC_CLK_CHANGE_TIMEOUT ((uint32_t)10000)`

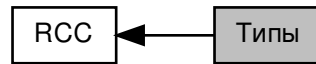
Время ожидания смены источника тактирования

См. определение в файле `niietcm4_rcc.h` строка 52

Используется в `RCC_WaitClkChange()`.

6.49 Типы

Граф связей класса Типы:



Структуры данных

- struct [RCC_PLLInit_TypeDef](#)
Структура инициализации PLL.

Макросы

- [#define IS_RCC_PLL_REF\(PLL_REF\)](#)
Макрос проверки аргументов типа [RCC_PLLRef_TypeDef](#).
- [#define IS_RCC_PLL_NO\(PLL_NO\)](#)
Макрос проверки аргументов типа [RCC_PLLNO_TypeDef](#).
- [#define IS_RCC_UART_CLK\(UART_CLK\)](#)
Макрос проверки аргументов типа [RCC_UARTClk_TypeDef](#).
- [#define IS_RCC_SPI_CLK\(SPI_CLK\)](#)
Макрос проверки аргументов типа [RCC_SPIClk_TypeDef](#).
- [#define IS_RCC_USB_CLK\(USB_CLK\)](#)
Макрос проверки аргументов типа [RCC_USBClk_TypeDef](#).
- [#define IS_RCC_USB_FREQ\(USB_FREQ\)](#)
Макрос проверки аргументов типа [RCC_USBFreq_TypeDef](#).
- [#define IS_RCC_ADC_CLK\(ADC_CLK\)](#)
Макрос проверки аргументов типа [RCC_ADCClk_TypeDef](#).
- [#define IS_RCC_SYS_CLK\(SYS_CLK\)](#)
Макрос проверки аргументов типа [RCC_SysClk_TypeDef](#).
- [#define IS_RCC_PERIPH_CLK\(PERIPH_CLK\)](#)
Макрос проверки аргументов типа [RCC_PeriphClk_TypeDef](#).
- [#define IS_RCC_PERIPH_RST\(PERIPH_RST\)](#)
Макрос проверки аргументов типа [RCC_PeriphRst_TypeDef](#).
- [#define IS_RCC_PLLDIV\(PLLDIV\) \(\(\(PLLDIV\) & \(\(uint32_t\)0xFFFFF00\)\) == \(\(uint32_t\)0x00\)\)](#)
Макрос проверки значения выходного делителя PLL на попадание в допустимый диапазон.
- [#define IS_RCC_PLL_NR\(PLL_NR\) \(\(\(PLL_NR\) <= \(\(uint32_t\)33\)\) && \(\(PLL_NR\) >= \(\(uint32_t\)2\)\)\)](#)
Макрос проверки значения опорного делителя PLL на попадание в допустимый диапазон.
- [#define IS_RCC_PLL_NF\(PLL_NF\) \(\(\(PLL_NF\) <= \(\(uint32_t\)513\)\) && \(\(PLL_NF\) >= \(\(uint32_t\)2\)\)\)](#)
Макрос проверки значения делителя ОС PLL на попадание в допустимый диапазон.
- [#define IS_RCC_CLK_DIV\(CLK_DIV\) \(\(CLK_DIV\) < \(\(uint32_t\)64\)\)](#)
Макрос проверки значения делителя тактового сигнала на попадание в допустимый диапазон.

- `#define IS_RCC_SYS_FREQ(SYS_FREQ) (((SYS_FREQ) < ((uint32_t)200000000)) && ((SYS_FREQ) >= ((uint32_t)1000000)))`

Макрос проверки значения желаемой частоты при автонастройке в допустимый диапазон.

Перечисления

- enum `RCC_PLLRef_TypeDef` { `RCC_PLLRef_XI_OSC`, `RCC_PLLRef_USB_CLK`, `RCC_PLLRef_USB_60MHz`, `RCC_PLLRef_ETH_25MHz` }

Выбор источника опорного сигнала PLL.

- enum `RCC_PLLNO_TypeDef` { `RCC_PLLNO_Disable`, `RCC_PLLNO_Div2`, `RCC_PLLNO_Div4 = 3` }

Выходной делитель NO.

- enum `RCC_UARTClk_TypeDef` { `RCC_UARTClk_SYSCLK`, `RCC_UARTClk_XI_OSC`, `RCC_UARTClk_USB_CLK`, `RCC_UARTClk_USB_60MHz` }

Выбор источника тактирования для UART.

- enum `RCC_SPIClk_TypeDef` { `RCC_SPIClk_SYSCLK`, `RCC_SPIClk_XI_OSC`, `RCC_SPIClk_USB_CLK`, `RCC_SPIClk_USB_60MHz` }

Выбор источника тактирования для SPI.

- enum `RCC_USBClk_TypeDef` { `RCC_USBClk_XI_OSC`, `RCC_USBClk_USB_CLK` }

Выбор источника тактирования для USB.

- enum `RCC_USBFreq_TypeDef` { `RCC_USBFreq_12MHz`, `RCC_USBFreq_24MHz` }

Выбор фиксированной частоты на входе CLK_USB.

- enum `RCC_ADCClk_TypeDef` { `RCC_ADCClk_0`, `RCC_ADCClk_1`, `RCC_ADCClk_2`, `RCC_ADCClk_3`, `RCC_ADCClk_4`, `RCC_ADCClk_5`, `RCC_ADCClk_6`, `RCC_ADCClk_7`, `RCC_ADCClk_8`, `RCC_ADCClk_9`, `RCC_ADCClk_10`, `RCC_ADCClk_11` }

Выбор модуля ADC для настройки его тактового сигнала.

- enum `RCC_SysClk_TypeDef` { `RCC_SysClk_CPE_Sel`, `RCC_SysClk_POR`, `RCC_SysClk_XI_OSC`, `RCC_SysClk_PLL`, `RCC_SysClk_PLLDIV`, `RCC_SysClk_USB60MHz`, `RCC_SysClk_USB_CLK`, `RCC_SysClk_ETH25MHz` }

Выбор источника системной частоты.

- enum `RCC_PeriphClk_TypeDef` { `RCC_PeriphClk_QEP0` = ((uint32_t)(1<<1)), `RCC_PeriphClk_QEP1` = ((uint32_t)(1<<2)), `RCC_PeriphClk_CMP` = ((uint32_t)(1<<9)), `RCC_PeriphClk_PWM0` = ((uint32_t)(1<<10)), `RCC_PeriphClk_PWM1` = ((uint32_t)(1<<11)), `RCC_PeriphClk_PWM2` = ((uint32_t)(1<<12)), `RCC_PeriphClk_PWM3` = ((uint32_t)(1<<13)), `RCC_PeriphClk_PWM4` = ((uint32_t)(1<<14)), `RCC_PeriphClk_PWM5` = ((uint32_t)(1<<15)), `RCC_PeriphClk_PWM6` = ((uint32_t)(1<<16)), `RCC_PeriphClk_PWM7` = ((uint32_t)(1<<17)), `RCC_PeriphClk_PWM8` = ((uint32_t)(1<<18)), `RCC_PeriphClk_WD` = ((uint32_t)(1<<19)), `RCC_PeriphClk_I2C0` = ((uint32_t)(1<<20)), `RCC_PeriphClk_I2C1` = ((uint32_t)(1<<21)), `RCC_PeriphClk_ADC` = ((uint32_t)(1<<24)) }

Управление тактированием периферийных блоков

- enum `RCC_PeriphRst_TypeDef` { `RCC_PeriphRst_WD` = ((uint32_t)(1<<0)), `RCC_PeriphRst_I2C0` = ((uint32_t)(1<<1)), `RCC_PeriphRst_I2C1` = ((uint32_t)(1<<2)), `RCC_PeriphRst_USB` = ((uint32_t)(1<<3)), `RCC_PeriphRst_Timer0` = ((uint32_t)(1<<4)), `RCC_PeriphRst_Timer1` = ((uint32_t)(1<<5)), `RCC_PeriphRst_Timer2` = ((uint32_t)(1<<6)), `RCC_PeriphRst_UART0` = ((uint32_t)(1<<7)), `RCC_PeriphRst_UART1` = ((uint32_t)(1<<8)), `RCC_PeriphRst_UART2` = ((uint32_t)(1<<9)), `RCC_PeriphRst_UART3` = ((uint32_t)(1<<10)), `RCC_PeriphRst_SPI0` =

```

((uint32_t)(1<<11)),
RCC_PeriphRst_SPI1 = ((uint32_t)(1<<12)), RCC_PeriphRst_SPI2 = ((uint32_t)(1<<13)),
RCC_PeriphRst_SPI3 = ((uint32_t)(1<<14)), RCC_PeriphRst_ETH = ((uint32_t)(1<<15)),
RCC_PeriphRst_QEP0 = ((uint32_t)(1<<0)), RCC_PeriphRst_QEP1 = ((uint32_t)(1<<1)),
RCC_PeriphRst_PWM0 = ((uint32_t)(1<<2)), RCC_PeriphRst_PWM1 = ((uint32_t)←
t)(1<<3)),
RCC_PeriphRst_PWM2 = ((uint32_t)(1<<4)), RCC_PeriphRst_PWM3 = ((uint32_t)←
t)(1<<5)), RCC_PeriphRst_PWM4 = ((uint32_t)(1<<6)), RCC_PeriphRst_PWM5 =
((uint32_t)(1<<7)),
RCC_PeriphRst_PWM6 = ((uint32_t)(1<<8)), RCC_PeriphRst_PWM7 = ((uint32_t)←
t)(1<<9)), RCC_PeriphRst_PWM8 = ((uint32_t)(1<<10)), RCC_PeriphRst_CAP0 =
((uint32_t)(1<<11)),
RCC_PeriphRst_CAP1 = ((uint32_t)(1<<12)), RCC_PeriphRst_CAP2 = ((uint32_t)←
t)(1<<13)), RCC_PeriphRst_CAP3 = ((uint32_t)(1<<14)), RCC_PeriphRst_CAP4 =
((uint32_t)(1<<15)),
RCC_PeriphRst_CAP5 = ((uint32_t)(1<<16)), RCC_PeriphRst_CMP = ((uint32_t)(1<<17))
}

```

Управление сбросом периферийных блоков

6.49.1 Подробное описание

6.49.2 Макросы

6.49.2.1 #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 203

Используется в `RCC_ADCClkCmd()` и `RCC_ADCClkDivConfig()`.

6.49.2.2 #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 273

Используется в `RCC_PeriphClkCmd()`.

6.49.2.3 `#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 99

Используется в `RCC_PLLInit()`.

6.49.2.4 `#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 78

Используется в `RCC_PLLAutoConfig()` и `RCC_PLLInit()`.

6.49.2.5 `#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 140

Используется в `RCC_SPIClkSel()`.

6.49.2.6 `#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 236

Используется в `RCC_SysClkSel()`.

6.49.2.7 #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 119

Используется в `RCC_UARTClkSel()`.

6.49.2.8 #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 159

Используется в `RCC_USBClkConfig()`.

6.49.2.9 #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 176

Используется в `RCC_USBClkConfig()`.

6.49.3 Перечисления

6.49.3.1 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

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 183

6.49.3.2 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 249

6.49.3.3 enum RCC_PeriphRst_TypeDef

Управление сбросом периферийных блоков

Элементы перечислений

RCC_PeriphRst_WD Управление сбросом сторожевого таймера
RCC_PeriphRst_I2C0 Управление сбросом блока I2C 0
RCC_PeriphRst_I2C1 Управление сбросом блока I2C 1
RCC_PeriphRst_USB Управление сбросом блока USB
RCC_PeriphRst_Timer0 Управление сбросом блока Timer 0
RCC_PeriphRst_Timer1 Управление сбросом блока Timer 1
RCC_PeriphRst_Timer2 Управление сбросом блока Timer 2
RCC_PeriphRst_UART0 Управление сбросом блока UART 0
RCC_PeriphRst_UART1 Управление сбросом блока UART 1
RCC_PeriphRst_UART2 Управление сбросом блока UART 2
RCC_PeriphRst_UART3 Управление сбросом блока UART 3
RCC_PeriphRst_SPI0 Управление сбросом блока SPI 0

RCC_PeriphRst_SPI1 Управление сбросом блока SPI 1
 RCC_PeriphRst_SPI2 Управление сбросом блока SPI 2
 RCC_PeriphRst_SPI3 Управление сбросом блока SPI 3
 RCC_PeriphRst_ETH Управление сбросом блока Ethernet
 RCC_PeriphRst_QEP0 Управление сбросом блока QEP 0
 RCC_PeriphRst_QEP1 Управление сбросом блока QEP 1
 RCC_PeriphRst_PWM0 Управление сбросом блока PWM 0
 RCC_PeriphRst_PWM1 Управление сбросом блока PWM 1
 RCC_PeriphRst_PWM2 Управление сбросом блока PWM 2
 RCC_PeriphRst_PWM3 Управление сбросом блока PWM 3
 RCC_PeriphRst_PWM4 Управление сбросом блока PWM 4
 RCC_PeriphRst_PWM5 Управление сбросом блока PWM 5
 RCC_PeriphRst_PWM6 Управление сбросом блока PWM 6
 RCC_PeriphRst_PWM7 Управление сбросом блока PWM 7
 RCC_PeriphRst_PWM8 Управление сбросом блока PWM 8
 RCC_PeriphRst_CAP0 Управление сбросом блока CAP 0
 RCC_PeriphRst_CAP1 Управление сбросом блока CAP 1
 RCC_PeriphRst_CAP2 Управление сбросом блока CAP 2
 RCC_PeriphRst_CAP3 Управление сбросом блока CAP 3
 RCC_PeriphRst_CAP4 Управление сбросом блока CAP 4
 RCC_PeriphRst_CAP5 Управление сбросом блока CAP 5
 RCC_PeriphRst_CMP Управление сбросом блока аналогового компаратора

См. определение в файле niietcm4_rcc.h строка 293

6.49.3.4 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 88

6.49.3.5 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 66

6.49.3.6 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 128

6.49.3.7 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 220

6.49.3.8 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 107

6.49.3.9 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 149

6.49.3.10 enum RCC_USBFreq_TypeDef

Выбор фиксированной частоты на входе CLK_USB.

Элементы перечислений

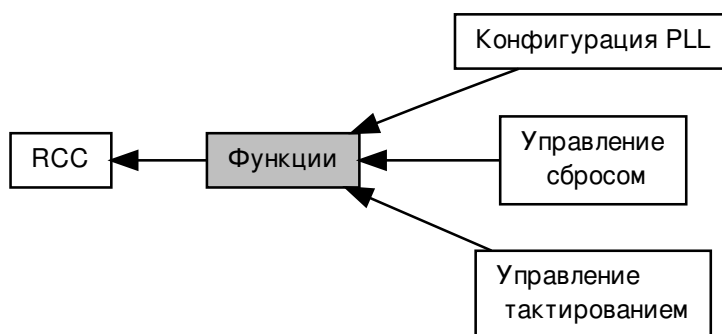
RCC_USBFreq_12MHz 12 МГц сигнал на входе CLK_USB

RCC_USBFreq_24MHz 24 МГц сигнал на входе CLK_USB

См. определение в файле niietcm4_rcc.h строка 166

6.50 Функции

Граф связей класса 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.

6.50.1 Подробное описание

6.50.2 Функции

6.50.2.1 `void RCC_SysClkDiv2Out (FunctionalState State)`

Включение генерации тактового сигнала с частой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState : <ul style="list-style-type: none">• <code>ENABLE</code> - переводит H[0] в выход включенной альтернативной функцией 2.• <code>DISABLE</code> - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

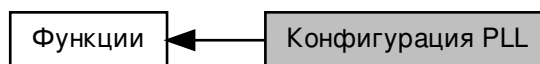
Нет

См. определение в файле `niietcm4_rcc.c` строка 106

Перекрестные ссылки `GPIO_InitTypeDef::GPIO_AltFunc`, `GPIO_AltFunc_2`, `GPIO_InitTypeDef::GPIO_Dir`, `GPIO_Dir_Out`, `GPIO_Init()`, `GPIO_InitTypeDef::GPIO_Mode`, `GPIO_Mode_AltFunc`, `GPIO_InitTypeDef::GPIO_Out`, `GPIO_Out_En`, `GPIO_InitTypeDef::GPIO_Pin`, `GPIO_Pin_0`, `GPIO_StructInit()` и `IS_FUNCTIONAL_STATE`.

6.51 Конфигурация PLL

Граф связей класса 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.

6.51.1 Подробное описание

6.51.2 Функции

6.51.2.1 `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 `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 <code>RCC_PLLRef_TypeDef</code> .
<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_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().

6.51.2.2 void RCC_PLLDeInit ()

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 292

Перекрестные ссылки RCC_PLL_CTRL_Reset_Value, RCC_PLL_NF_Reset_Value, RCC_PLLNR_Reset_Value и RCC_PLL_OD_Reset_Value.

6.51.2.3 void RCC_PLLInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Инициализирует PLL согласно параметрам структуры RCC_PLLInit_Struct.

Значение выходной частоты PLL вычисляется с использованием значений опорного NR и выходного NO делителей, а также делителя обратной связи NF по формуле:

$$F_{OUT} = (F_{IN} \times NF) / (NO \times NR),$$

где F_{IN} – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

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

где частота фазового детектора F_{REF} вычисляется по формуле:

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

а частота F_{VCO} вычисляется по формуле:

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

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 256

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()`.

6.51.2.4 `void RCC_PLLPowerDownCmd (FunctionalState State)`

Управление режимом PowerDown PLL.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 307

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

6.51.2.5 `void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)`

Заполнение каждого члена структуры `RCC_PLLInit_Struct` значениями по умолчанию.

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
-------------------------	---

Возвращаемые значения

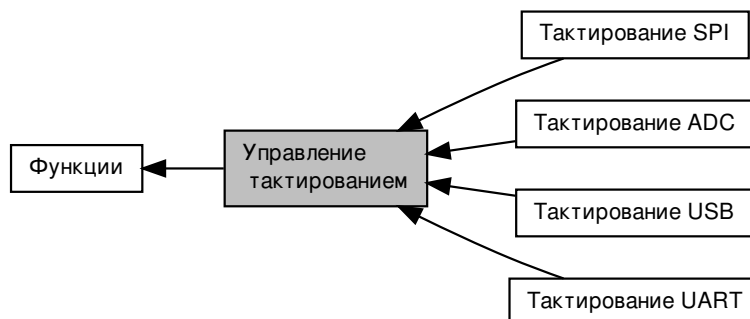
Нет

См. определение в файле `niietcm4_rcc.c` строка 278

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`.

6.52 Управление тактированием

Граф связей класса 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.

6.52.1 Подробное описание

6.52.2 Функции

6.52.2.1 `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_Periph↵ Clk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_↵ TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 336

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_RCC_PERIPH_CLK.

6.52.2.2 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_Type↵ Def .
------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 359

Перекрестные ссылки IS_RCC_SYS_CLK и RCC_WaitClkChange().

Используется в RCC_PLLAutoConfig().

6.52.2.3 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

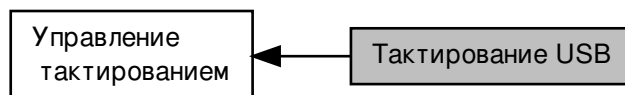
Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле niietcm4_rcc.c строка 388

6.53 Тактирование USB

Граф связей класса 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.

6.53.1 Подробное описание

6.53.2 Функции

6.53.2.1 void RCC_USBCLKCmd (FunctionalState State)

Включение тактирования USB.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 419

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.53.2.2 void RCC_USBCLKConfig (RCC_USBCLK_TypeDef RCC_USBCLK, RCC_USBFreq_TypeDef RCC_USBFreq)

Настройка источника тактового сигнала для USB.

Аргументы

RCC_USBCLK	Выбор источника тактирования. Параметр принимает любое значение из RCC_USBCLK_TypeDef .
RCC_USBFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

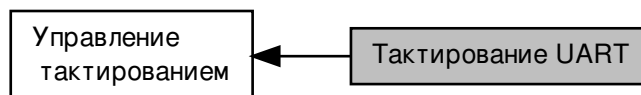
Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 402

Перекрестные ссылки IS_RCC_USB_CLK и IS_RCC_USB_FREQ.

6.54 Тактирование UART

Граф связей класса 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.

6.54.1 Подробное описание

6.54.2 Функции

6.54.2.1 void [RCC_UARTClkCmd](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.54.2.2 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 Functional State .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 482

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_UART_ALL_PERIPH.

6.54.2.3 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_UARTClk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

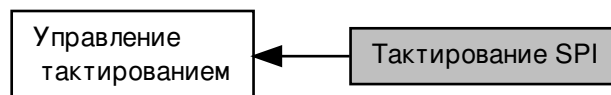
Нет

См. определение в файле niietcm4_rcc.c строка 442

Перекрестные ссылки IS_RCC_UART_CLK и IS_UART_ALL_PERIPH.

6.55 Тактирование SPI

Граф связей класса 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.

6.55.1 Подробное описание

6.55.2 Функции

6.55.2.1 void [RCC_SPIClkCmd](#) (NT_SPI_TypeDef * SPIx, [FunctionalState](#) State)

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 642

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_SPI_ALL_PERIPH.

6.55.2.2 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 Functional State .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 604

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_SPI_ALL_PERIPH.

6.55.2.3 void RCC_SPIClkSel (NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

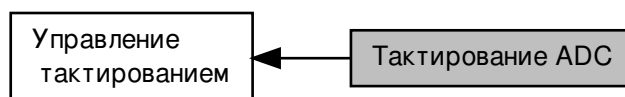
Нет

См. определение в файле niietcm4_rcc.c строка 563

Перекрестные ссылки IS_RCC_SPI_CLK и IS_SPI_ALL_PERIPH.

6.56 Тактирование ADC

Граф связей класса 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.

6.56.1 Подробное описание

6.56.2 Функции

6.56.2.1 void [RCC_ADCClkCmd](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, [FunctionalState](#) State)

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 773

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](#).

6.56.2.2 void [RCC_ADCClkDivConfig](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, uint32_t DivVal, [FunctionalState](#) DivState)

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

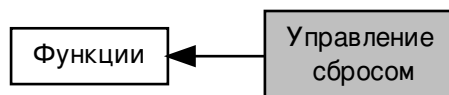
Нет

См. определение в файле niietcm4_rcc.c строка 689

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.

6.57 Управление сбросом

Граф связей класса Управление сбросом:



Функции

- void [RCC_PeriphRstCmd](#) ([RCC_PeriphRst_TypeDef](#) RCC_PeriphRst, [FunctionalState](#) State)
Вывод из состояния сброса периферийных блоков.

6.57.1 Подробное описание

6.57.2 Функции

6.57.2.1 void [RCC_PeriphRstCmd](#) ([RCC_PeriphRst_TypeDef](#) RCC_PeriphRst, [FunctionalState](#) State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_PeriphRst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 863

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_RCC_PERIPH_RST](#), [RCC_PeriphRst_Enum](#), [RCC_PeriphRst_I2C0](#), [RCC_PeriphRst_I2C1](#), [RCC_PeriphRst_SPI0](#), [RCC_PeriphRst_SPI1](#), [RCC_PeriphRst_SPI2](#), [RCC_PeriphRst_SPI3](#), [RCC_PeriphRst_Timer0](#), [RCC_PeriphRst_Timer1](#), [RCC_PeriphRst_Timer2](#), [RCC_PeriphRst_UART0](#), [RCC_PeriphRst_UART1](#), [RCC_PeriphRst_UART2](#), [RCC_PeriphRst_UART3](#), [RCC_PeriphRst_USB](#) и [RCC_PeriphRst_WD](#).

Используется в [UART_DeInit\(\)](#).

6.58 Типы

Граф связей класса Типы:



Структуры данных

- struct `RTC_Time_TypeDef`
Структура времени.
- struct `RTC_Date_TypeDef`
Структура даты.

Макросы

- `#define IS_RTC_PSECOND(PSECOND) ((PSECOND) <= 0x3FF)`
Макрос проверки попадания значений долей секунд в допустимый диапазон.
- `#define IS_RTC_SECOND(SECOND) ((SECOND) <= 59)`
Макрос проверки попадания значений секунд в допустимый диапазон.
- `#define IS_RTC_MINUTE(MINUTE) ((MINUTE) <= 59)`
Макрос проверки попадания значений минут в допустимый диапазон.
- `#define IS_RTC_HOUR(HOUR) ((HOUR) <= 23)`
Макрос проверки попадания значений часов в допустимый диапазон.
- `#define IS_RTC_WEEKDAY(WEEKDAY)`
Макрос проверки аргументов типа `RTC_Weekday_TypeDef`.
- `#define IS_RTC_DAY(DAY) (((DAY) > 0) && ((DAY) <= 31))`
Макрос проверки попадания значений дней в допустимый диапазон.
- `#define IS_RTC_MONTH(MONTH)`
Макрос проверки аргументов типа `RTC_Month_TypeDef`.
- `#define IS_RTC_YEAR(YEAR) ((YEAR) <= 99)`
Макрос проверки попадания значений лет в допустимый диапазон.
- `#define IS_RTC_FORMAT(FORMAT)`
Макрос проверки аргументов типа `RTC_Format_TypeDef`.

Перечисления

- enum `RTC_Weekday_TypeDef` {
`RTC_Weekday_Monday = ((uint32_t)0x01), RTC_Weekday_Tuesday = ((uint32_t)0x02), R↵`
`TC_Weekday_Wednesday = ((uint32_t)0x03), RTC_Weekday_Thursday = ((uint32_t)0x04),`
`RTC_Weekday_Friday = ((uint32_t)0x05), RTC_Weekday_Saturday = ((uint32_t)0x06), R↵`
`TC_Weekday_Sunday = ((uint32_t)0x07) }`
Дни недели.

- enum `RTC_Month_TypeDef` {
`RTC_Month_January` = ((uint32_t)0x01), `RTC_Month_February` = ((uint32_t)0x02), `RTC_Month_March` = ((uint32_t)0x03), `RTC_Month_April` = ((uint32_t)0x04),
`RTC_Month_May` = ((uint32_t)0x05), `RTC_Month_June` = ((uint32_t)0x06), `RTC_Month_July` = ((uint32_t)0x07), `RTC_Month_August` = ((uint32_t)0x08),
`RTC_Month_September` = ((uint32_t)0x09), `RTC_Month_October` = ((uint32_t)0x10), `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.

6.58.1 Подробное описание

6.58.2 Макросы

6.58.2.1 #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

6.58.2.2 #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

6.58.2.3 #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

6.58.3 Перечисления

6.58.3.1 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

6.58.3.2 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

6.58.3.3 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

6.59 Функции

Граф связей класса Функции:



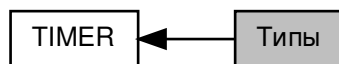
Функции

- 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)

6.59.1 Подробное описание

6.60 Типы

Граф связей класса 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.

6.60.1 Подробное описание

6.60.2 Макросы

6.60.2.1 `#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()`.

6.60.3 Перечисления

6.60.3.1 `enum TIMER_ExtInput_TypeDef`

Настройка внешнего тактирования таймера.

Элементы перечислений

`TIMER_ExtInput_Disable` Внешнее тактирование не используется.

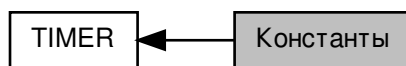
`TIMER_ExtInput_CountClk` Таймер считает по внешнему тактовому сигналу.

`TIMER_ExtInput_CountEn` Таймер считает по внутреннему тактовому сигналу и только тогда, когда на выводе "1".

См. определение в файле nietscm4_timer.h строка 56

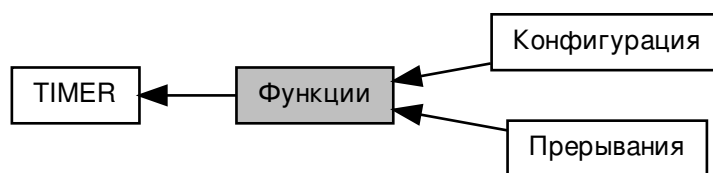
6.61 Константы

Граф связей класса Константы:



6.62 Функции

Граф связей класса Функции:



Группы

- [Конфигурация](#)
- [Прерывания](#)

6.62.1 Подробное описание

6.63 Конфигурация

Граф связей класса 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.

6.63.1 Подробное описание

6.63.2 Функции

6.63.2.1 void TIMER_Cmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_timer.c` строка 65

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_TIMER_ALL_PERIPH`.

```
6.63.2.2 void TIMER_ExtInputConfig ( NT_TIMER_TypeDef * TIMEx,
    TIMER_ExtInput_TypeDef TIMER_ExtInput )
```

Выбор режима работы входа внешнего тактирования.

Аргументы

<code>TIMEx</code>	Выбор таймера, где x лежит в диапазоне 0-2.
<code>TIMER_ExtInput</code>	Выбор режима работы. Параметр принимает любое значение из TIMER_ExtInput_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_timer.c` строка 171

Перекрестные ссылки `IS_TIMER_ALL_PERIPH`, `IS_TIMER_EXT_INPUT`, `TIMER_ExtInputCountClk` и `TIMER_ExtInput_CountEn`.

```
6.63.2.3 void TIMER_FreqConfig ( NT_TIMER_TypeDef * TIMEx, 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

<code>TIMEx</code>	Выбор таймера, где x лежит в диапазоне 0-2.
<code>TimerClkFreq</code>	Частота в Гц, которой тактируется таймер.
<code>TimerFreq</code>	Частота опустошения таймера в Гц.

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_timer.c` строка 102

Перекрестные ссылки `IS_TIMER_ALL_PERIPH`.

```
6.63.2.4 uint32_t TIMER_GetCounter ( NT_TIMER_TypeDef * TIMEx )
```

Получение текущего значения счетчика.

Аргументы

<code>TIMEx</code>	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`.

6.63.2.5 `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`.

6.63.2.6 `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`.

6.63.2.7 `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.

6.63.2.8 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

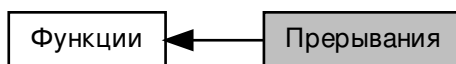
Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 116

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.64 Прерывания

Граф связей класса 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.

6.64.1 Подробное описание

6.64.2 Функции

6.64.2.1 void [TIMER_ITCmd](#) (NT_TIMER_TypeDef * TIMERx, [FunctionalState](#) State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 200

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#) и [IS_TIMER_ALL_PERIPH](#).

6.64.2.2 [FlagStatus](#) [TIMER_ITStatus](#) (NT_TIMER_TypeDef * TIMERx)

Чтение статуса прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

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

См. определение в файле niietcm4_timer.c строка 214

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.64.2.3 void TIMER_ITStatusClear (NT_TIMER_TypeDef * TIMERx)

Очищение статусного бита прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
--------	---

Возвращаемые значения

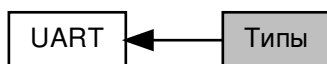
Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 238

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.65 Типы

Граф связей класса Типы:



Структуры данных

- struct `UART_ModemInit_TypeDef`
Структура инициализации модемного режима.
- struct `UART_Init_TypeDef`
Структура инициализации UART.

Макросы

- `#define IS_UART_INT_DIV(INT_DIV) (((INT_DIV) > ((uint32_t)0x0)) && ((INT_DIV) < ((uint32_t)0x10000)))`
Макрос проверки соответствия величины целой части делителя baudrate UART диапазону.
- `#define IS_UART_FRAC_DIV(FRAC_DIV) ((FRAC_DIV) < ((uint32_t)0x40))`
Макрос проверки соответствия величины дробной части делителя baudrate UART диапазону.
- `#define IS_UART_DATA(DATA) ((DATA) < ((uint32_t)0x100))`
Макрос проверки корректности передаваемых данных.
- `#define IS_UART_FLAG(FLAG)`
Макрос проверки аргументов типа `UART_Flag_Typedef`.
- `#define IS_UART_ERROR(ERROR)`
Макрос проверки аргументов типа `UART_Error_Typedef`.
- `#define IS_UART_IT_SOURCE(IT_SOURCE) (((IT_SOURCE) > ((uint32_t)0x0)) && ((IT_SOURCE) < ((uint32_t)0x800)))`
Макрос проверки аргументов типа `UART_ITSource_Typedef`.
- `#define IS_UART_GET_IT_SOURCE(IT_SOURCE)`
Макрос проверки номера пина при работе с пинами по отдельности.
- `#define IS_UART_DIR(DIR)`
Макрос проверки аргументов типа `UART_Dir_Typedef`.
- `#define IS_UART_STOP_BIT(STOP_BIT)`
Макрос проверки аргументов типа `UART_StopBit_TypeDef`.
- `#define IS_UART_PARITY_BIT(PARITY_BIT)`
Макрос проверки аргументов типа `UART_ParityBit_TypeDef`.
- `#define IS_UART_DATA_WIDTH(DATA_WIDTH)`
Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.
- `#define IS_UART_FIFO_LEVEL(FIFO_LEVEL)`
Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

Перечисления

- enum `UART_Flag_Typedef` {
`UART_Flag_InvCTS`, `UART_Flag_InvDSR`, `UART_Flag_InvDCD`, `UART_Flag_Busy`,
`UART_Flag_RxFIFOEmpty`, `UART_Flag_TxFIFOFull`, `UART_Flag_RxFIFOFull`, `UART_↵`
`Flag_TxFIFOEmpty`,
`UART_Flag_InvRI` }
Перечень флагов.
- enum `UART_Error_Typedef` { `UART_Error_Frame`, `UART_Error_Parity`, `UART_Error_↵`
`Break`, `UART_Error_Overflow` }
Перечень ошибок приемника.
- enum `UART_ITSource_Typedef` {
`UART_ITSource_ChangeRI` = ((uint32_t)0x00000001), `UART_ITSource_ChangeCTS` =
((uint32_t)0x00000002), `UART_ITSource_ChangeDCD` = ((uint32_t)0x00000004), `UART_↵`
`ITSource_ChangeDSR` = ((uint32_t)0x00000008),
`UART_ITSource_RxFIFOLevel` = ((uint32_t)0x00000010), `UART_ITSource_TxFIFOLevel` =
((uint32_t)0x00000020), `UART_ITSource_RecieveTimeout` = ((uint32_t)0x00000040), `UART_↵`
`ITSource_ErrorFrame` = ((uint32_t)0x00000080),
`UART_ITSource_ErrorParity` = ((uint32_t)0x00000100), `UART_ITSource_ErrorBreak` =
((uint32_t)0x00000200), `UART_ITSource_ErrorOverflow` = ((uint32_t)0x00000400) }
Источники прерываний UART.
- enum `UART_Dir_Typedef` { `UART_Dir_Rx`, `UART_Dir_Tx` }
Направления передачи UART.
- enum `UART_StopBit_TypeDef` { `UART_StopBit_1`, `UART_StopBit_2` }
Выбор режима передачи стопового бита.
- enum `UART_ParityBit_TypeDef` {
`UART_ParityBit_Disable`, `UART_ParityBit_Odd`, `UART_ParityBit_Even`, `UART_Parity_↵`
`Bit_High`,
`UART_ParityBit_Low` }
Выбор режима бита четности.
- enum `UART_DataWidth_TypeDef` { `UART_DataWidth_5`, `UART_DataWidth_6`, `UART_↵`
`DataWidth_7`, `UART_DataWidth_8` }
Количество передаваемых/принимаемых информационных бит.
- enum `UART_FIFOLevel_TypeDef` {
`UART_FIFOLevel_1_8`, `UART_FIFOLevel_1_4`, `UART_FIFOLevel_1_2`, `UART_FIFO_↵`
`Level_3_4`,
`UART_FIFOLevel_7_8` }
Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

6.65.1 Подробное описание

6.65.2 Макросы

6.65.2.1 #define IS_UART_DATA_WIDTH(DATA_WIDTH)

Макроопределение:

```
((DATA_WIDTH) == UART_DataWidth_5) || \
((DATA_WIDTH) == UART_DataWidth_6) || \
((DATA_WIDTH) == UART_DataWidth_7) || \
((DATA_WIDTH) == UART_DataWidth_8))
```

Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.

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

Используется в `UART_Init()`.

6.65.2.2 #define IS_UART_DIR(DIR)

Макроопределение:

```
((DIR) == UART_Dir_Rx) || \
((DIR) == UART_Dir_Tx)
```

Макрос проверки аргументов типа `UART_Dir_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 177

Используется в `UART_DMACmd()` и `UART_ITFIFOLevelConfig()`.

6.65.2.3 #define IS_UART_ERROR(ERROR)

Макроопределение:

```
((ERROR) == UART_Error_Frame) || \
((ERROR) == UART_Error_Parity) || \
((ERROR) == UART_Error_Break) || \
((ERROR) == UART_Error_Overflow)
```

Макрос проверки аргументов типа `UART_Error_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 117

Используется в `UART_ErrorStatus()`.

6.65.2.4 #define IS_UART_FIFO_LEVEL(FIFO_LEVEL)

Макроопределение:

```
((FIFO_LEVEL) == UART_FIFOLevel_1_8) || \
((FIFO_LEVEL) == UART_FIFOLevel_1_4) || \
((FIFO_LEVEL) == UART_FIFOLevel_1_2) || \
((FIFO_LEVEL) == UART_FIFOLevel_3_4) || \
((FIFO_LEVEL) == UART_FIFOLevel_7_8))
```

Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

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

Используется в `UART_Init()` и `UART_ITFIFOLevelConfig()`.

6.65.2.5 #define IS_UART_FLAG(FLAG)

Макроопределение:

```
((FLAG) == UART_Flag_InvCTS) || \
((FLAG) == UART_Flag_InvDSR) || \
((FLAG) == UART_Flag_InvDCD) || \
((FLAG) == UART_Flag_Busy) || \
((FLAG) == UART_Flag_RxFIFOEmpty) || \
((FLAG) == UART_Flag_TxFIFOFull) || \
((FLAG) == UART_Flag_RxFIFOFull) || \
((FLAG) == UART_Flag_TxFIFOEmpty) || \
((FLAG) == UART_Flag_InvRI))
```

Макрос проверки аргументов типа `UART_Flag_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 91

Используется в `UART_FlagStatus()`.

6.65.2.6 #define IS_UART_GET_IT_SOURCE(IT_SOURCE)

Макроопределение:

```
((IT_SOURCE) == UART_ITSource_ChangeRI) || \
    ((IT_SOURCE) == \
    UART_ITSource_ChangeCTS) || \
    ((IT_SOURCE) == \
    UART_ITSource_ChangeDCD) || \
    ((IT_SOURCE) == \
    UART_ITSource_ChangeDSR) || \
    ((IT_SOURCE) == \
    UART_ITSource_RxFIFOLevel) || \
    ((IT_SOURCE) == \
    UART_ITSource_TxFIFOLevel) || \
    ((IT_SOURCE) == \
    UART_ITSource_RecieveTimeout) || \
    ((IT_SOURCE) == \
    UART_ITSource_ErrorFrame) || \
    ((IT_SOURCE) == \
    UART_ITSource_ErrorParity) || \
    ((IT_SOURCE) == \
    UART_ITSource_ErrorBreak) || \
    ((IT_SOURCE) == \
    UART_ITSource_ErrorOverflow))
```

Макрос проверки номера пина при работе с пинами по отдельности.

См. определение в файле niietcm4_uart.h строка 151

Используется в UART_ITMaskedStatus() и UART_ITRawStatus().

6.65.2.7 #define IS_UART_PARITY_BIT(PARITY_BIT)

Макроопределение:

```
((PARITY_BIT) == UART_ParityBit_Disable) || \
    ((PARITY_BIT) == UART_ParityBit_Odd) || \
    ((PARITY_BIT) == UART_ParityBit_Even) || \
    ((PARITY_BIT) == UART_ParityBit_High) || \
    ((PARITY_BIT) == UART_ParityBit_Low))
```

Макрос проверки аргументов типа UART_ParityBit_TypeDef.

См. определение в файле niietcm4_uart.h строка 214

Используется в UART_Init().

6.65.2.8 #define IS_UART_STOP_BIT(STOP_BIT)

Макроопределение:

```
((STOP_BIT) == UART_StopBit_1) || \
    ((STOP_BIT) == UART_StopBit_2))
```

Макрос проверки аргументов типа UART_StopBit_TypeDef.

См. определение в файле niietcm4_uart.h строка 194

Используется в UART_Init().

6.65.3 Перечисления

6.65.3.1 enum UART_DataWidth_TypeDef

Количество передаваемых/принимаемых информационных бит.

Элементы перечислений

UART_DataWidth_5 Длина информационного слова 5 бит.

UART_DataWidth_6 Длина информационного слова 6 бит.

UART_DataWidth_7 Длина информационного слова 7 бит.

UART_DataWidth_8 Длина информационного слова 8 бит.

См. определение в файле niietcm4_uart.h строка 224

6.65.3.2 enum UART_Dir_Typedef

Направления передачи UART.

Элементы перечислений

UART_Dir_Rx Передача.

UART_Dir_Tx Прием.

См. определение в файле niietcm4_uart.h строка 167

6.65.3.3 enum UART_Error_Typedef

Перечень ошибок приемника.

Элементы перечислений

UART_Error_Frame Флаг ошибки в структуре кадра.

UART_Error_Parity Флаг ошибки контроля четности.

UART_Error_Break Флаг разрыва линии.

UART_Error_Overflow Флаг переполнения буфера приемника.

См. определение в файле niietcm4_uart.h строка 105

6.65.3.4 enum UART_FIFOLevel_TypeDef

Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

Элементы перечислений

UART_FIFOLevel_1_8 Заполнение FIFO на 1/8.

UART_FIFOLevel_1_4 Заполнение FIFO на 1/4.

UART_FIFOLevel_1_2 Заполнение FIFO на 1/2.

UART_FIFOLevel_3_4 Заполнение FIFO на 3/4.

UART_FIFOLevel_7_8 Заполнение FIFO на 7/8.

См. определение в файле niietcm4_uart.h строка 246

6.65.3.5 enum UART_Flag_Typedef

Перечень флагов.

Элементы перечислений

UART_Flag_InvCTS Флаг инверсии сигнала на линии UART_CTS.
 UART_Flag_InvDSR Флаг инверсии сигнала на линии UART_DSR.
 UART_Flag_InvDCD Флаг инверсии сигнала на линии UART_DSR.
 UART_Flag_Busy Флаг занятости блока UART.
 UART_Flag_RxFIFOEmpty Флаг пустоты буфера приемника.
 UART_Flag_TxFIFOFull Флаг заполнения буфера передатчика.
 UART_Flag_RxFIFOFull Флаг заполнения буфера приемника.
 UART_Flag_TxFIFOEmpty Флаг пустоты буфера передатчика.
 UART_Flag_InvRI Флаг инверсии сигнала на линии UART_RI.

См. определение в файле niietcm4_uart.h строка 74

6.65.3.6 enum UART_ITSource_Typedef

Источники прерываний UART.

Элементы перечислений

UART_ITSource_ChangeRI Изменение состояния линии UART_RI
 UART_ITSource_ChangeCTS Изменение состояния линии UART_CTS
 UART_ITSource_ChangeDCD Изменение состояния линии UART_DCD
 UART_ITSource_ChangeDSR Изменение состояния линии UART_DSR
 UART_ITSource_RxFIFOLevel Порог переполнения буфера приемника
 UART_ITSource_TxFIFOLevel Порог опустошения буфера передатчика
 UART_ITSource_RecieveTimeout Таймаут приема данных
 UART_ITSource_ErrorFrame Ошибка в структуре кадра
 UART_ITSource_ErrorParity Ошибка контроля четности
 UART_ITSource_ErrorBreak Разрыв линии
 UART_ITSource_ErrorOverflow Переполнение буфера приемника

См. определение в файле niietcm4_uart.h строка 126

6.65.3.7 enum UART_ParityBit_TypeDef

Выбор режима бита четности.

Элементы перечислений

UART_ParityBit_Disable Не передается, не проверяется.
 UART_ParityBit_Odd Проверка нечетности данных.
 UART_ParityBit_Even Проверка четности данных.
 UART_ParityBit_High Бит четности постоянно равен единице.
 UART_ParityBit_Low Бит четности постоянно равен нулю.

См. определение в файле niietcm4_uart.h строка 201

6.65.3.8 enum UART_StopBit_TypeDef

Выбор режима передачи стопового бита.

Элементы перечислений

UART_StopBit_1 Один стоповый бит.

UART_StopBit_2 Два стоповых бита.

См. определение в файле niietcm4_uart.h строка 184

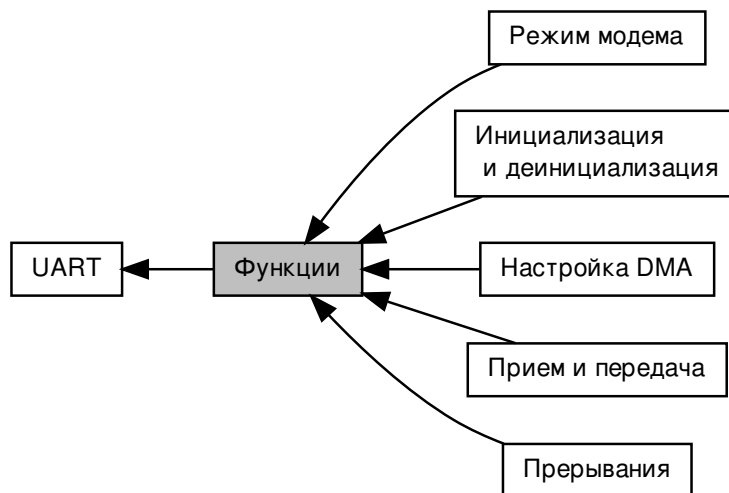
6.66 Константы

Граф связей класса Константы:



6.67 Функции

Граф связей класса 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.

6.67.1 Подробное описание

6.67.2 Функции

6.67.2.1 void [UART_BaudRateDivConfig](#) (NT_UART_TypeDef * UARTx, uint32_t IntDiv,
uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

6.67.2.2 void UART_Break (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.67.2.3 void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.68 Инициализация и деинициализация

Граф связей класса 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.

6.68.1 Подробное описание

6.68.2 Функции

6.68.2.1 void UART_DeInit (NT_UART_TypeDef * UARTx)

Устанавливает все регистры UART значениями по умолчанию.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 120

Перекрестные ссылки IS_UART_ALL_PERIPH, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_UART2, RCC_PeriphRst_UART3 и RCC_PeriphRstCmd().

6.68.2.2 OperationStatus UART_Init (NT_UART_TypeDef * UARTx, UART_Init_TypeDef * UART_InitStruct)

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле `niietcm4_uart.c` строка 159

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_UART_ALL_PERIPH`, `IS_UART_DATA_WIDTH`, `IS_UART_FIFO_LEVEL`, `IS_UART_PARITY_BIT`, `IS_UART_STOP_BIT`, `UART_Init_TypeDef::UART_BaudRate`, `UART_Init_TypeDef::UART_ClkFreq`, `UART_Init_TypeDef::UART_DataWidth`, `UART_Init_TypeDef::UART_FIFOEn`, `UART_Init_TypeDef::UART_FIFOLevelRx`, `UART_Init_TypeDef::UART_FIFOLevelTx`, `UART_Init_TypeDef::UART_ParityBit`, `UART_ParityBit_Even`, `UART_ParityBit_High`, `UART_ParityBit_Low`, `UART_ParityBit_Odd`, `UART_Init_TypeDef::UART_RxEn`, `UART_Init_TypeDef::UART_StopBit` и `UART_Init_TypeDef::UART_TxEn`.

6.68.2.3 `void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)`

Заполнение каждого члена структуры `UART_InitStruct` значениями по умолчанию.

Аргументы

UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

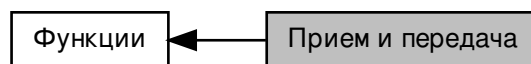
Нет

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

Перекрестные ссылки `EXT_OSC_VALUE`, `UART_Init_TypeDef::UART_BaudRate`, `UART_Init_TypeDef::UART_ClkFreq`, `UART_Init_TypeDef::UART_DataWidth`, `UART_DataWidth_8`, `UART_Init_TypeDef::UART_FIFOEn`, `UART_FIFOLevel_1_2`, `UART_Init_TypeDef::UART_FIFOLevelRx`, `UART_Init_TypeDef::UART_FIFOLevelTx`, `UART_Init_TypeDef::UART_ParityBit`, `UART_ParityBit_Disable`, `UART_Init_TypeDef::UART_RxEn`, `UART_Init_TypeDef::UART_StopBit`, `UART_StopBit_1` и `UART_Init_TypeDef::UART_TxEn`.

6.69 Прием и передача

Граф связей класса Прием и передача:



Функции

- void [UART_SendData](#) (NT_UART_TypeDef *UARTx, uint32_t Data)
Передача слова данных.
- uint32_t [UART_RecieveData](#) (NT_UART_TypeDef *UARTx)
Прием слова данных.
- [FlagStatus UART_FlagStatus](#) (NT_UART_TypeDef *UARTx, [UART_Flag_Typedef UART_↵_Flag](#))
Запрос состояния выбранного флага.
- [FlagStatus UART_ErrorStatus](#) (NT_UART_TypeDef *UARTx, [UART_Error_Typedef UAR↵_Error](#))
Запрос состояния выбранного флага ошибки.
- void [UART_ErrorStatusClear](#) (NT_UART_TypeDef *UARTx)
Очистка флагов ошибки.

6.69.1 Подробное описание

6.69.2 Функции

6.69.2.1 [FlagStatus UART_ErrorStatus](#) (NT_UART_TypeDef * UARTx, [UART_Error_Typedef UART_Error](#))

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error↵_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

6.69.2.2 void [UART_ErrorStatusClear](#) (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.69.2.3 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_Typedef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

6.69.2.4 uint32_t UART_RecieveData (NT_UART_TypeDef * UARTx)

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.69.2.5 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

6.70 Режим модема

Граф связей класса 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.

6.70.1 Подробное описание

6.70.2 Функции

6.70.2.1 void [UART_ModemConfig](#) (NT_UART_TypeDef * UARTx, [UART_ModemInit_TypeDef](#) * [UART_ModemInitStruct](#))

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.

Аргументы

UART_ModemInitStruct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
----------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

6.70.2.2 void [UART_ModemStructInit](#) ([UART_ModemInit_TypeDef](#) * [UART_ModemInitStruct](#))

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
--------------------------------	--

Возвращаемые значения

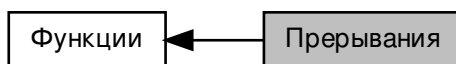
Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки [UART_ModemInit_TypeDef::UART_CTSEn](#), [UART_ModemInit_TypeDef::UART_InvDTR](#), [UART_ModemInit_TypeDef::UART_InvRTS](#) и [UART_ModemInit_TypeDef::UART_RTSEn](#).

6.71 Прерывания

Граф связей класса Прерывания:



Функции

- void [UART_ITFIFOLevelConfig](#) (NT_UART_TypeDef *UARTx, [UART_Dir_Typedef](#) UART_Dir, [UART_FIFOLevel_TypeDef](#) UART_FIFOLevel)
Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.
- void [UART_ITCmd](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource, [FunctionalState](#) State)
Маскирование выбранных прерываний.
- [FlagStatus](#) [UART_ITRawStatus](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource)
Запрос немаскированного состояния прерывания.
- [FlagStatus](#) [UART_ITMaskedStatus](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource)
Запрос маскированного состояния прерывания.
- void [UART_ITStatusClear](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_Typedef](#) UART_ITSource)
Сброс флагов состояния выбранных прерываний.

6.71.1 Подробное описание

6.71.2 Функции

6.71.2.1 void [UART_ITCmd](#) (NT_UART_TypeDef * UARTx, [UART_ITSource_Typedef](#) UART_ITSource, [FunctionalState](#) State)

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 410

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

6.71.2.2 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 буффером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFOLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и UART_Dir_Rx.

6.71.2.3 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерывания. Параметр принимает любое значение из UART_ITSource_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

6.71.2.4 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерывания. Параметр принимает любое значение из UART_ITSource_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

6.71.2.5 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любое значение из UART_ITSource_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

6.72 Настройка DMA

Граф связей класса 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_DMACmd](#) (NT_UART_TypeDef *UARTx, [UART_Dir_Typedef](#) UART_Dir, [FunctionalState](#) State)
Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

6.72.1 Подробное описание

6.72.2 Функции

6.72.2.1 void [UART_DMABlkOnErrCmd](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#) и [IS_UART_ALL_PERIPH](#).

6.72.2.2 void [UART_DMACmd](#) (NT_UART_TypeDef * UARTx, [UART_Dir_Typedef](#) UART_Dir, [FunctionalState](#) State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

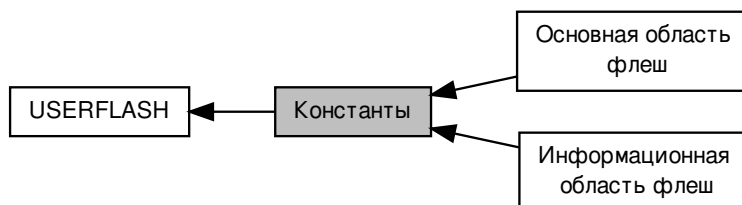
Нет

См. определение в файле `niietcm4_uart.c` строка 520

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_UART_ALL_PERIPH`, `IS_UART_DIR` и `UART_Dir_Rx`.

6.73 Константы

Граф связей класса Константы:



Группы

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

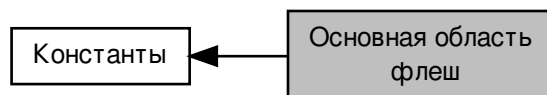
Макросы

- `#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.

6.73.1 Подробное описание

6.74 Основная область флеш

Граф связей класса 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.

6.74.1 Подробное описание

6.74.2 Макросы

6.74.2.1 `#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()`.

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

Общее количество страниц.

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

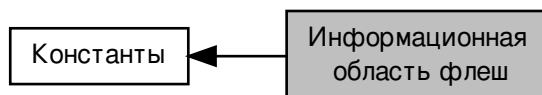
6.74.2.3 `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`

Общий размер основной области.

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

6.75 Информационная область флеш

Граф связей класса 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.

6.75.1 Подробное описание

6.75.2 Макросы

6.75.2.1 `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES`

Размер страницы в байтах.

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

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

Общее количество страниц.

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

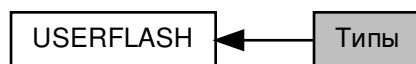
6.75.2.3 `#define USERFLASH_INFO_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`

Общий размер информационной области.

См. определение в файле `niietcm4_userflash.h` строка 88

6.76 Типы

Граф связей класса 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.

6.76.1 Подробное описание

6.76.2 Макросы

6.76.2.1 `#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

6.76.3 Перечисления

6.76.3.1 `enum USERFLASH_Status_TypeDef`

Статус работы контроллера пользовательской флеш-памяти.

Элементы перечислений

`USERFLASH_Status_None` Операция выполняется или отсутствует.

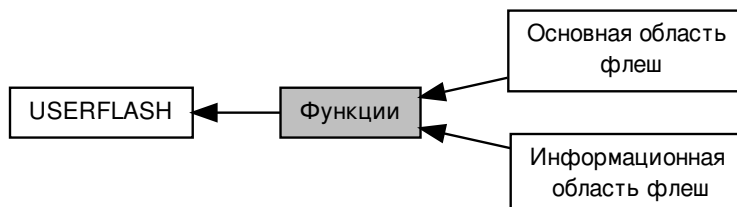
`USERFLASH_Status_Complete` Операция успешно завершена.

`USERFLASH_Status_Error` Операция завершена с ошибкой.

См. определение в файле `niietcm4_userflash.h` строка 112

6.77 Функции

Граф связей класса 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.

6.77.1 Подробное описание

6.77.2 Функции

6.77.2.1 void USERFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера пользовательской флеш.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 66

6.77.2.2 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.77.2.3 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

6.77.2.4 void USERFLASH_OperationStatusClear ()

Очищает статус работы контроллера пользовательской флэш.

Возвращаемые значения

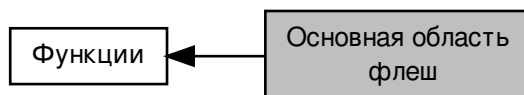
Нет.

См. определение в файле niietcm4_userflash.c строка 93

Используется в USERFLASH_Info_Read() и USERFLASH_Read().

6.78 Основная область флеш

Граф связей класса 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.

6.78.1 Подробное описание

6.78.2 Функции

6.78.2.1 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](#).

6.78.2.2 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.

6.78.2.3 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.

6.78.2.4 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

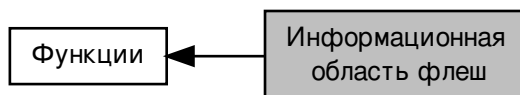
Нет

См. определение в файле niietcm4_userflash.c строка 148

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.79 Информационная область флеш

Граф связей класса 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.

6.79.1 Подробное описание

6.79.2 Функции

6.79.2.1 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.

6.79.2.2 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.

6.79.2.3 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	
-----	--

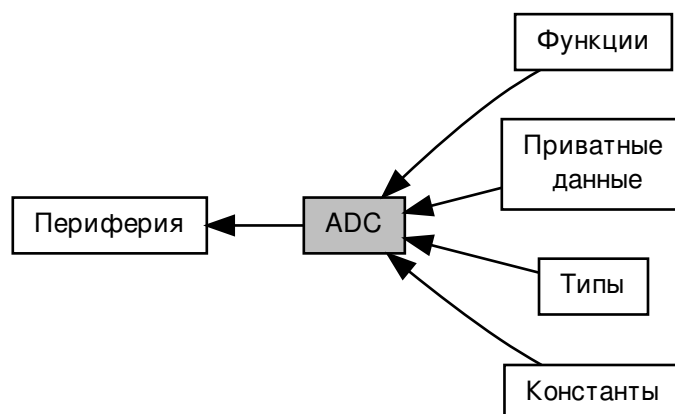
См. определение в файле niietcm4_userflash.c строка 206

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.80 ADC

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

Граф связей класса ADC:



Группы

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

6.80.1 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](#).

Перед началом работы с АЦП необходимо записать во все поля выбора канала, подключаемого к цифровому компаратору, запрещенное значение 0x18-0x1F. Это связано с тем, что после сброса к компараторам подключается нулевой канал, который будучи неиспользованным, (например, секвенсор измеряет по 5 и 6 каналам) приводит к зависанию секвенсоров. 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. Отсутствие задержки перезапуска в асинхронном режиме приводит к гарантированному зависанию секвенсоров.

Рекомендуется использовать только один секвенсор, либо несколько секвенсоров, но работающих только синхронно (один источник запуска, одинаковые задержки перезапуска). Любая другая асинхронная конфигурация секвенсоров имеет тенденцию к зависанию. Особенность аппаратная, программно не обходится. Чтобы восстановить работу АЦП после зависания необходимо сделать полный аппаратный сброс микроконтроллера.

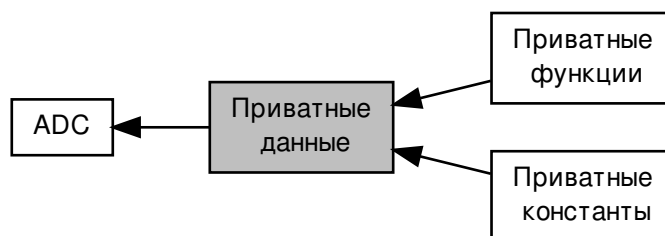
Общий вид процесса инициализации:

- 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.

6.81 Приватные данные

Граф связей класса Приватные данные:



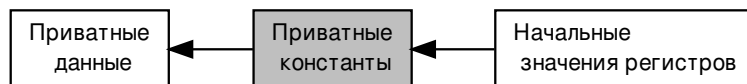
Группы

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

6.81.1 Подробное описание

6.82 Приватные константы

Граф связей класса Приватные константы:



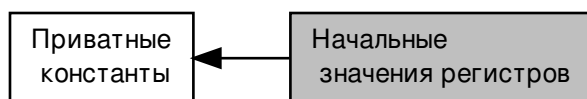
Группы

- [Начальные значения регистров](#)

6.82.1 Подробное описание

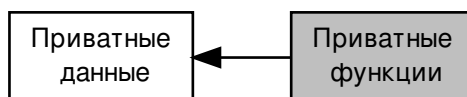
6.83 Начальные значения регистров

Граф связей класса Начальные значения регистров:



6.84 Приватные функции

Граф связей класса 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.

6.84.1 Подробное описание

6.84.2 Функции

6.84.2.1 void ADC_Cmd (ADC_Module_TypeDef ADC_Module, FunctionalState State)

Включение модуля АЦП.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 115

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

6.84.2.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1024

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

6.84.2.3 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 307

Перекрестные ссылки IS_ADC_DC_MODULE.

6.84.2.4 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1040

Перекрестные ссылки IS_ADC_DC_MODULE.

6.84.2.5 void ADC_DC_Init (ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры [ADC_DC_InitStruct](#).

Аргументы

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

См. определение в файле niietcm4_adc.c строка 326

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](#).

6.84.2.6 void ADC_DC_ITCmd (ADC_DC_Module_TypeDef ADC_DC_Module,
FunctionalState State)

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 596

Перекрестные ссылки ADC_DC_ITGenCmd(), ADC_DC_ITMaskCmd(), IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

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

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 620

Перекрестные ссылки IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_MODULE.

```
6.84.2.8 void ADC_DC_ITGenCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                               FunctionalState State )
```

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 553

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

6.84.2.9 void ADC_DC_ITMaskCmd (ADC_DC_Module_TypeDef ADC_DC_Module,
FunctionalState State)

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 570

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

6.84.2.10 FlagStatus ADC_DC_ITMaskedStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов маскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 662

Перекрестные ссылки IS_ADC_DC_MODULE.

6.84.2.11 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 637

Перекрестные ссылки IS_ADC_DC_MODULE.

6.84.2.12 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 688

Перекрестные ссылки IS_ADC_DC_MODULE.

6.84.2.13 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 349

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.

6.84.2.14 FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1058

Перекрестные ссылки IS_ADC_DC_MODULE.

6.84.2.15 void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

Сброс выходного триггера цифрового компаратора.

Внимание

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1086

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

6.84.2.16 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 130

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.

6.84.2.17 void ADC_Init (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef * ADC_InitStruct)

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

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

См. определение в файле niietcm4_adc.c строка 206

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.

6.84.2.18 void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение секвенсора.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
----------------	---

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 855

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

6.84.2.19 void ADC_SEQ_DeInit (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 365

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.

6.84.2.20 void ADC_SEQ_DMAMCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 496

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

6.84.2.21 void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 479

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

6.84.2.22 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 512

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.23 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 537

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.24 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 983

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.25 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1008

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.26 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 943

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.27 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 968

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.28 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 905

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.29 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Получение результата измерений из буфера секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 887

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.30 uint32_t ADC_SEQ_GetFIFOLoad (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 923

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.31 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 757

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
6.84.2.32 void ADC_SEQ_Init ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,  
ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct )
```

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 425

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](#).

```
6.84.2.33 void ADC_SEQ_ITCmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                             FunctionalState State )
```

Включение прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 704

Перекрестные ссылки [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

```
6.84.2.34 void ADC_SEQ_ITConfig ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                              uint32_t ADC_SEQ_ITRate, FunctionalState ADC_SEQ_ITCountSEQRst )
```

Настройка вызова прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 732

Перекрестные ссылки [IS_ADC_SEQ_IT_RATE](#), [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

6.84.2.35 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 775

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.36 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 814

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.37 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 789

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.38 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 839

Перекрестные ссылки IS_ADC_SEQ_MODULE.

6.84.2.39 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

ADC_SEQ_InitStruct	Указатель на структуру типа ADC_SEQ_Init_TypeDef , которую необходимо проинициализировать.
--------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 460

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.

6.84.2.40 void ADC_SEQ_SWReq ()

Программный запуск измерений всех разрешенных секвенсоров.

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 875

6.84.2.41 void ADC_StructInit (ADC_Init_TypeDef * ADC_InitStruct)

Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

Аргументы

ADC_InitStruct	Указатель на структуру типа ADC_Init_TypeDef , которую необходимо проинициализировать.
----------------	--

Возвращаемые значения

Нет

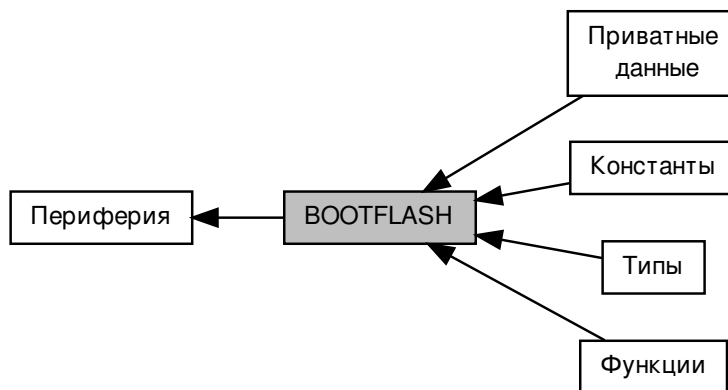
См. определение в файле niietcm4_adc.c строка 290

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.

6.85 BOOTFLASH

Драйвер для загрузочной флеш-памяти.

Граф связей класса BOOTFLASH:



Группы

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

6.85.1 Подробное описание

Драйвер для загрузочной флеш-памяти.

Внимание

Для контроллера K1921BK01T необходимо вызывать функции записи и стирания только из другой функции, расположенной в ОЗУ.

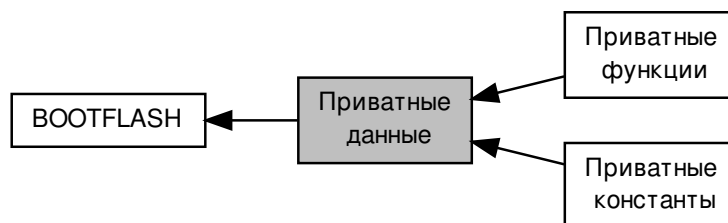
Общий вид процесса инициализации:

- 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.

6.86 Приватные данные

Граф связей класса Приватные данные:



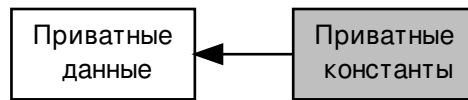
Группы

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

6.86.1 Подробное описание

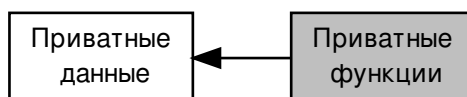
6.87 Приватные константы

Граф связей класса Приватные константы:



6.88 Приватные функции

Граф связей класса 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.

6.88.1 Подробное описание

6.88.2 Функции

6.88.2.1 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](#).

6.88.2.2 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.

6.88.2.3 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.

6.88.2.4 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

6.88.2.5 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

--

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.88.2.6 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_bootflash.c строка 88

6.88.2.7 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 102

6.88.2.8 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.

6.88.2.9 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	
-----	--

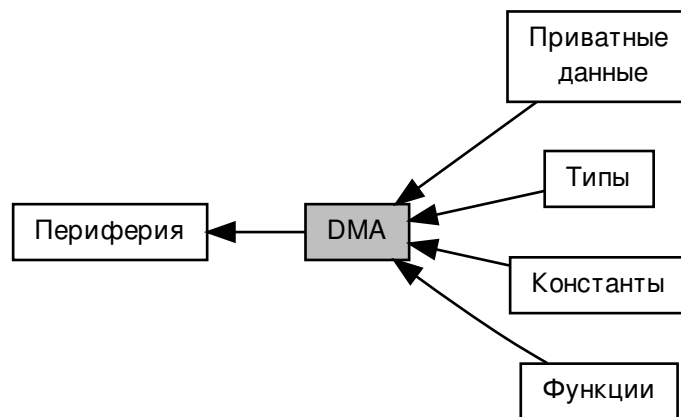
См. определение в файле niietcm4_bootflash.c строка 128

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

6.89 DMA

Драйвер для работы с контроллером прямого доступа памяти.

Граф связей класса DMA:



Группы

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

6.89.1 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.

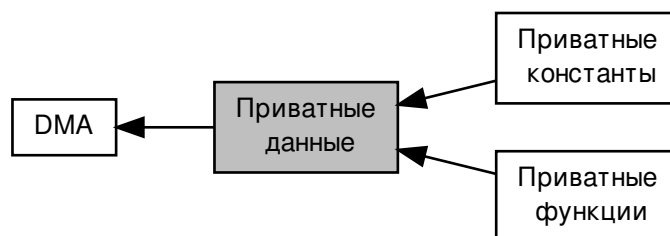
Общий вид процесса инициализации:

- 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.

6.90 Приватные данные

Граф связей класса Приватные данные:



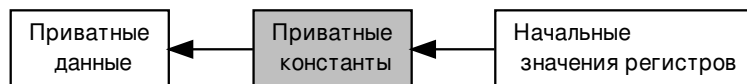
Группы

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

6.90.1 Подробное описание

6.91 Приватные константы

Граф связей класса Приватные константы:



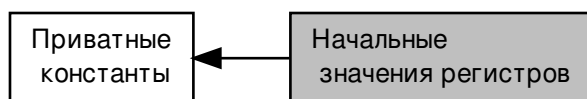
Группы

- [Начальные значения регистров](#)

6.91.1 Подробное описание

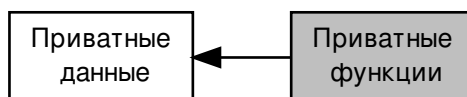
6.92 Начальные значения регистров

Граф связей класса Начальные значения регистров:



6.93 Приватные функции

Граф связей класса 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.

6.93.1 Подробное описание

6.93.2 Функции

6.93.2.1 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

6.93.2.2 void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)

Деинициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 87

Перекрестные ссылки [DMA_Channel_TypeDef::CHANNEL_CFG](#), [DMA_Channel_TypeDef::DST←_DATA_END](#) и [DMA_Channel_TypeDef::SRC_DATA_END](#).

6.93.2.3 void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)

Активация каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 373

Перекрестные ссылки [IS_DMA_CHANNEL](#) и [IS_FUNCTIONAL_STATE](#).

Используется в [DMA_Init\(\)](#).

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

Инициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 102

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Channel_TypeDef::CHANNEL_CFG_bit, _CHANNEL_CFG_bits::CYCLE_CTRL, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, DMA_ChannelInit_TypeDef::DMA_DstDataInc, DMA_ChannelInit_TypeDef::DMA_DstDataSize, DMA_ChannelInit_TypeDef::DMA_DstProtect, DMA_ChannelInit_TypeDef::DMA_Mode, DMA_ChannelInit_TypeDef::DMA_NextUseburst, DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr, DMA_ChannelInit_TypeDef::DMA_SrcDataInc, DMA_ChannelInit_TypeDef::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_TypeDef::DMA_TransfersTotal, DMA_Channel_TypeDef::DST_DATA_END, _CHANNEL_CFG_bits::DST_INC, _CHANNEL_CFG_bits::DST_PROT_BUFFERABLE, _CHANNEL_CFG_bits::DST_PROT_CACHEABLE, _CHANNEL_CFG_bits::DST_PROT_PRIVILEGED, _CHANNEL_CFG_bits::DST_SIZE, IS_DMA_ARBITRATION_RATE, IS_DMA_DATA_INC, IS_DMA_DATA_SIZE, IS_DMA_MODE, IS_DMA_TRANSFERS_TOTAL, IS_FUNCTIONAL_STATE, _CHANNEL_CFG_bits::N_MINUS_1, _CHANNEL_CFG_bits::NEXT_USEBURST, DMA_Protect_TypeDef::PRIVELGED, _CHANNEL_CFG_bits::R_POWER, DMA_Channel_TypeDef::SRC_DATA_END, _CHANNEL_CFG_bits::SRC_INC, _CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE, _CHANNEL_CFG_bits::SRC_PROT_CACHEABLE, _CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED и _CHANNEL_CFG_bits::SRC_SIZE.

6.93.2.5 void DMA_ChannelStructInit (DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct)

Заполнение каждого члена структуры DMA_ChannelInitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 146

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

6.93.2.6 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 524

6.93.2.7 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 174

6.93.2.8 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 503

6.93.2.9 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

6.93.2.10 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

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

Аргументы

DMA_Init↔ Struct	Указатель на структуру типа DMA_Init_TypeDef , которая содержит конфигурационную информацию.
---------------------	--

Возвращаемые значения

Нет

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

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

6.93.2.11 `void DMA_MasterEnableCmd (FunctionalState State)`

Разрешения работы контроллера DMA.

Внимание

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

Аргументы

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

Возвращаемые значения

Нет

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

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#).

6.93.2.12 `FunctionalState DMA_MasterEnableStatus ()`

Состояние контроллера DMA.

Возвращаемые значения

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

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

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

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

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

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.93.2.14 `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 DMA_Protect_TypeDef .
---------------------	--

Возвращаемые значения

Нет

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

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

Используется в `DMA_Init()`.

6.93.2.15 `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 Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

6.93.2.16 `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

6.93.2.17 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

Аргументы

DMA_InitStruct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
----------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 212

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Init_TypeDef::DMA_Channel, DMA_Init_TypeDef::DMA_ChannelEnable, DMA_Init_TypeDef::DMA_HighPriority, DMA_Init_TypeDef::DMA_PrmAlt, DMA_Init_TypeDef::DMA_Protection, DMA_Init_TypeDef::DMA_ReqMask, DMA_Init_TypeDef::DMA_UseBurst и DMA_Protect_TypeDef::PRIVELGED.

6.93.2.18 void DMA_SWRequestCmd (uint32_t DMA_Channel)

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 308

Перекрестные ссылки IS_DMA_CHANNEL.

6.93.2.19 void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)

Установка пакетного обмена каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 324

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

6.93.2.20 FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

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

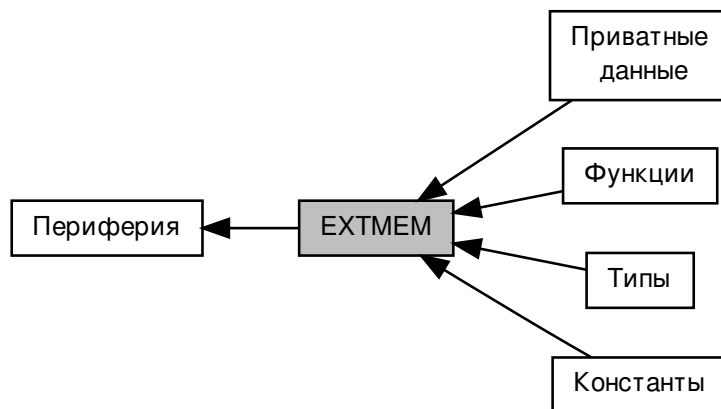
См. определение в файле niietcm4_dma.c строка 478

Перекрестные ссылки IS_GET_DMA_CHANNEL.

6.94 EXTMEM

Драйвер для интерфейса внешней памяти.

Граф связей класса EXTMEM:



Группы

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

6.94.1 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](#).

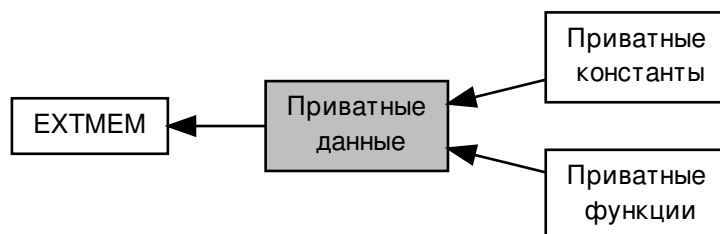
Общий вид процесса инициализации:

- 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.

6.95 Приватные данные

Граф связей класса Приватные данные:



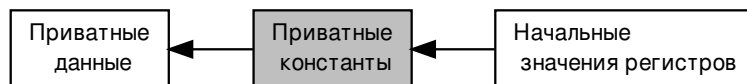
Группы

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

6.95.1 Подробное описание

6.96 Приватные константы

Граф связей класса Приватные константы:



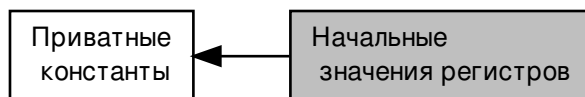
Группы

- [Начальные значения регистров](#)

6.96.1 Подробное описание

6.97 Начальные значения регистров

Граф связей класса Начальные значения регистров:



Макросы

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

6.97.1 Подробное описание

6.97.2 Макросы

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

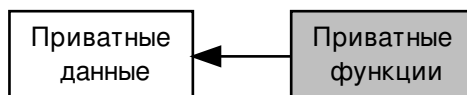
Значение по сбросу регистра EXT_MEM_CFG

См. определение в файле niietcm4_extmem.c строка 64

Используется в EXTMEM_DeInit().

6.98 Приватные функции

Граф связей класса 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.

6.98.1 Подробное описание

6.98.2 Функции

6.98.2.1 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.

6.98.2.2 void EXTMEM_Init (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Инициализирует внешнюю память согласно параметрам структуры EXTMEM_InitStruct.

Аргументы

EXTMEM_↔ InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_extmem.c строка 85

Перекрестные ссылки IS_EXTMEM_CE_MASK, IS_EXTMEM_READ_WAITSTATE, IS_EXTMEM_RW_WAITSTATE, IS_EXTMEM_WIDTH и IS_EXTMEM_WRITE_WAITSTATE.

6.98.2.3 void EXTMEM_StructInit (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Заполнение каждого члена структуры EXTMEM_InitStruct значениями по умолчанию.

Аргументы

EXTMEM_↔ InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
------------------------	---

Возвращаемые значения

Нет

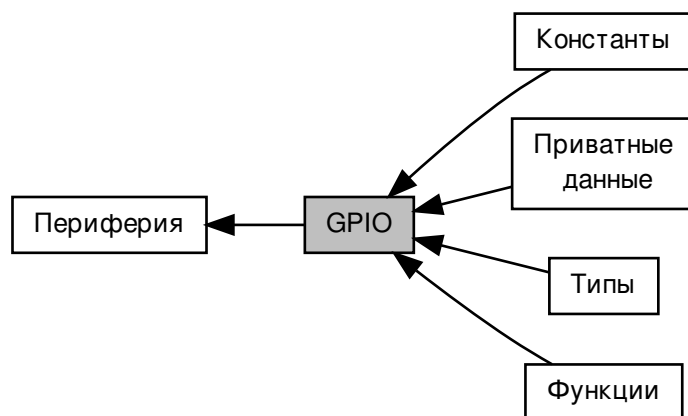
См. определение в файле niietcm4_extmem.c строка 107

Перекрестные ссылки EXTMEM_Init_TypeDef::CEMask, EXTMEM_Init_TypeDef::EXTMEM_↔ReadWaitState, EXTMEM_ReadWaitState_8, EXTMEM_Init_TypeDef::EXTMEM_RWWaitState, EXTMEM_RWWaitState_1, EXTMEM_Init_TypeDef::EXTMEM_Width, EXTMEM_Width_↔16bit, EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState и EXTMEM_WriteWaitState_1.

6.99 GPIO

Драйвер для управления портами ввода-вывода.

Граф связей класса GPIO:



Группы

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

6.99.1 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.

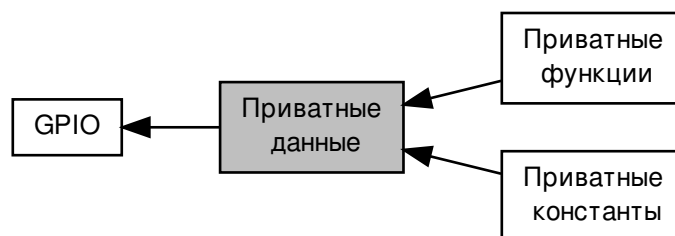
Общий вид процесса инициализации:

- 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.

6.100 Приватные данные

Граф связей класса Приватные данные:



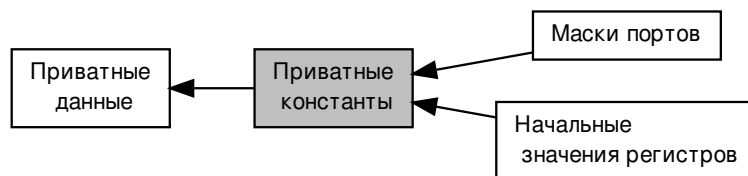
Группы

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

6.100.1 Подробное описание

6.101 Приватные константы

Граф связей класса Приватные константы:



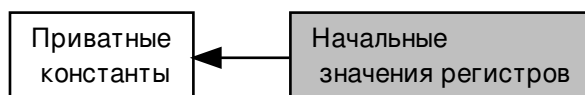
Группы

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

6.101.1 Подробное описание

6.102 Начальные значения регистров

Граф связей класса 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_GPIODCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIODSCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOPCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOEx_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)`

6.102.1 Подробное описание

6.102.2 Макросы

6.102.2.1 `#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()`.

6.102.2.2 `#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()`.

6.102.2.3 `#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()`.

6.102.2.4 `#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()`.

6.102.2.5 `#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()`.

6.102.2.6 `#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()`.

6.102.2.7 `#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()`.

6.102.2.8 `#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()`.

6.102.2.9 `#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()`.

6.102.2.10 `#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()`.

6.102.2.11 `#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()`.

6.102.2.12 `#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()`.

6.102.2.13 `#define GPIO_GPIOSEx_Reset_Value ((uint32_t)0x00000000)`

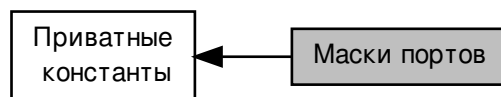
Значение по сбросу регистра GPIOSEx

См. определение в файле `niietcm4_gpio.c` строка 80

Используется в `GPIO_DeInit()`.

6.103 Маски портов

Граф связей класса 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)`

6.103.1 Подробное описание

6.103.2 Макросы

6.103.2.1 `#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()`.

6.103.2.2 `#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()`.

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

Маска регистров для порта GPIOA

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

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

Маска регистров для порта GPIOB

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

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

Маска регистров для порта GPIOC

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

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

Маска регистров для порта GPIOD

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

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

Маска регистров для порта GPIOE

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

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

Маска регистров для порта GPIOF

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

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

Маска регистров для порта GPIOG

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

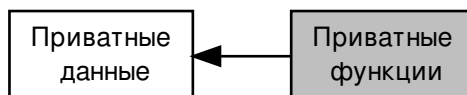
6.103.2.10 `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

Маска регистров для порта GPIOH

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

6.104 Приватные функции

Граф связей класса Приватные функции:



Функции

- void **GPIO_DeInit** (NT_GPIO_TypeDef *GPIOx)
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- void **GPIO_AltFuncConfig** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **GPIO_AltFunc_TypeDef** GPIO_AltFunc)
- void **GPIO_Init** (NT_GPIO_TypeDef *GPIOx, **GPIO_Init_TypeDef** *GPIO_InitStruct)
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- void **GPIO_StructInit** (**GPIO_Init_TypeDef** *GPIO_InitStruct)
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.
- uint32_t **GPIO_ReadBit** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Чтение состояния выбранного пина.
- uint32_t **GPIO_Read** (NT_GPIO_TypeDef *GPIOx)
Чтение состояния выбранного порта GPIOx.
- uint32_t **GPIO_ReadMask** (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)
Чтение состояния выбранного порта GPIOx с использованием маски.
- void **GPIO_WriteBit** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **BitAction** BitVal)
Изменение состояния выбранного пина.
- void **GPIO_Write** (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx.
- void **GPIO_WriteMask** (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx с использованием маски.
- void **GPIO_SetBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Установка выбранных пинов.
- void **GPIO_ClearBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Сброс выбранных пинов.
- void **GPIO_ToggleBits** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Переключение выбранных пинов в противоположное состояние.
- void **GPIO_QualConfig** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **GPIO_QualMode_TypeDef** Mode, uint32_t SamplePeriod)
Настройка фильтра выбранных пинов.
- void **GPIO_QualCmd** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **FunctionalState** State)
Включение входных фильтров.
- void **GPIO_SyncCmd** (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, **FunctionalState** State)
Включение пересинхронизации входов через 2 триггера-защелки.

- void [GPIO_ITConfig](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_IntType_TypeDef](#) IntType, [GPIO_IntPol_TypeDef](#) IntPol)
Настройка прерываний пинов.
- void [GPIO_ITCmd](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение прерываний выбранных пинов.
- void [GPIO_ITStatusClear](#) (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Очистка флагов прерываний выбранных пинов.

6.104.1 Подробное описание

6.104.2 Функции

6.104.2.1 void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 626

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.104.2.2 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.

6.104.2.3 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_Init↔ Struct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

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

Перекрестные ссылки [GPIO_Init_TypeDef::GPIO_AltFunc](#), [GPIO_Init_TypeDef::GPIO_Dir](#), [GPIO_Dir_In](#), [GPIO_Dir_Out](#), [GPIO_Init_TypeDef::GPIO_Load](#), [GPIO_Load_16mA](#), [GPIO_Load_8mA](#), [GPIO_Init_TypeDef::GPIO_Mode](#), [GPIO_Mode_AltFunc](#), [GPIO_Mode_IO](#), [GPIO_Init_TypeDef::GPIO_Out](#), [GPIO_Out_Dis](#), [GPIO_Out_En](#), [GPIO_Init_TypeDef::GPIO_OutMode](#), [GPIO_OutMode_OD](#), [GPIO_OutMode_PP](#), [GPIO_Init_TypeDef::GPIO_Pin](#), [GPIO_Init_TypeDef::GPIO_PullUp](#), [GPIO_PullUp_Dis](#), [GPIO_PullUp_En](#), [IS_GPIO_ALL_PERIPH](#), [IS_GPIO_ALT_FUNC](#), [IS_GPIO_DIR](#), [IS_GPIO_LOAD](#), [IS_GPIO_MODE](#), [IS_GPIO_OUT](#), [IS_GPIO_OUT_MODE](#), [IS_GPIO_PIN](#) и [IS_GPIO_PULLUP](#).

Используется в `RCC_SysClkDiv2Out()`.

```
6.104.2.4 void GPIO_ITCmd ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,
                          FunctionalState State )
```

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 913

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_GPIO_ALL_PERIPH](#) и [IS_GPIO_PIN](#).

```
6.104.2.5 void GPIO_ITConfig ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,
                             GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol )
```

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 876

Перекрестные ссылки GPIO_IntPol_Neg, GPIO_IntPol_Pos, GPIO_IntType_Edge, GPIO_IntType_Level, IS_GPIO_ALL_PERIPH, IS_GPIO_INT_POL, IS_GPIO_INT_TYPE и IS_GPIO_PIN.

6.104.2.6 void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 938

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.104.2.7 void GPIO_QualCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.104.2.8 void GPIO_QualConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_↵_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 664

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

6.104.2.9 uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx

См. определение в файле niietcm4_gpio.c строка 503

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.104.2.10 uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина

См. определение в файле niietcm4_gpio.c строка 477

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN и IS_GPIO_ALL_PERIPH.

6.104.2.11 uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)

Чтение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски чтения.

Возвращаемые значения

Состояние порта GPIOx с учетом маски	
---	--

См. определение в файле niietcm4_gpio.c строка 518

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.104.2.12 void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 609

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.104.2.13 void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)

Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 456

Перекрестные ссылки GPIO_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().

6.104.2.14 void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 814

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.104.2.15 void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Переключение выбранных пинов в противоположное состояние.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 643

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

6.104.2.16 void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.104.2.17 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 546

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

6.104.2.18 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

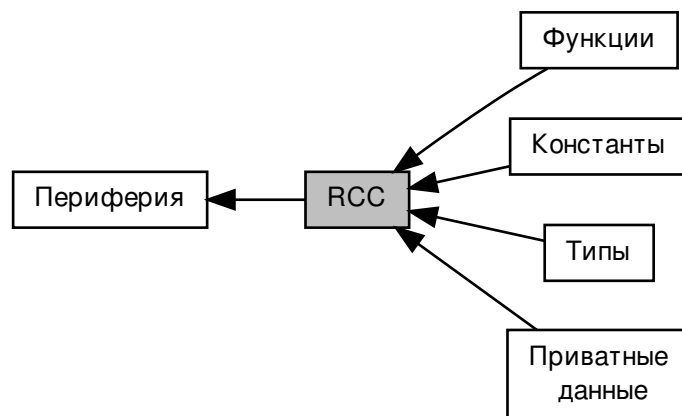
См. определение в файле niietcm4_gpio.c строка 586

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

6.105 RCC

Драйвер для работы с тактированием и сбросом периферийных блоков.

Граф связей класса RCC:



Группы

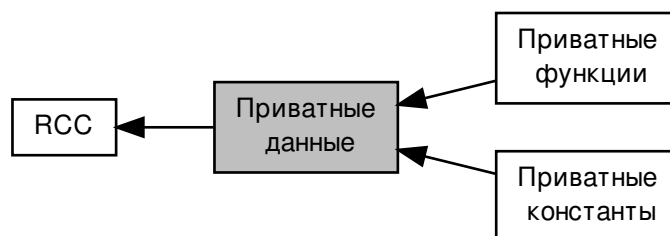
- [Константы](#)
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.105.1 Подробное описание

Драйвер для работы с тактированием и сбросом периферийных блоков.

6.106 Приватные данные

Граф связей класса Приватные данные:



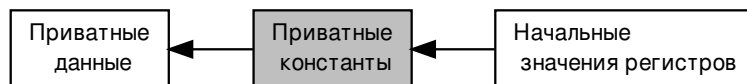
Группы

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

6.106.1 Подробное описание

6.107 Приватные константы

Граф связей класса Приватные константы:



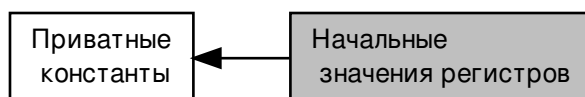
Группы

- [Начальные значения регистров](#)

6.107.1 Подробное описание

6.108 Начальные значения регистров

Граф связей класса 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)`

6.108.1 Подробное описание

6.108.2 Макросы

6.108.2.1 `#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().

6.108.2.2 `#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().

6.108.2.3 `#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().

6.108.2.4 `#define RCC_PLL_OD_Reset_Value ((uint32_t)0x00000000)`

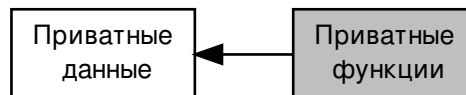
Значение по сбросу регистра PLL_OD

См. определение в файле niietcm4_rcc.c строка 55

Используется в RCC_PLLDeInit().

6.109 Приватные функции

Граф связей класса 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 SysClkFreq)`
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.

6.109.1 Подробное описание

6.109.2 Функции

6.109.2.1 void [RCC_ADCClkCmd](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, [FunctionalState](#) State)

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 773

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](#).

6.109.2.2 void [RCC_ADCClkDivConfig](#) ([RCC_ADCClk_TypeDef](#) RCC_ADCClk, uint32_t DivVal, [FunctionalState](#) DivState)

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.

DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .
----------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 689

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.

6.109.2.3 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_PeriphClk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 336

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_RCC_PERIPH_CLK.

6.109.2.4 void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_PeriphRst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 863

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_PERIPH_RST, RCC_PeriphRst_ETH, RCC_PeriphRst_I2C0, RCC_PeriphRst_I2C1, RCC_PeriphRst_SPI0, RCC_PeriphRst_SPI1, RCC_PeriphRst_SPI2, RCC_PeriphRst_SPI3, RCC_PeriphRst_Timer0, RCC_PeriphRst_Timer1,

RCC_PeriphRst_Timer2, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_UART2, RCC_PeriphRst_UART3, RCC_PeriphRst_USB и RCC_PeriphRst_WD.

Используется в UART_DeInit().

6.109.2.5 OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)

Автоматическая конфигурация PLL для получения желаемой системной частоты.

С учетом данных об источнике частоты для PLL, а также о значении желаемой частоты, вычисляются все необходимые коэффициенты.

Внимание

Если Freq < 50 МГц, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

RCC_PLLRef	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из RCC_PLLRef_TypeDef .
SysFreq	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 142

Перекрестные ссылки EXT_OSC_VALUE, IS_RCC_PLL_REF, IS_RCC_SYS_FREQ, RCC_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().

6.109.2.6 void RCC_PLLDeInit ()

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 292

Перекрестные ссылки RCC_PLL_CTRL_Reset_Value, RCC_PLL_NF_Reset_Value, RCC_PLLNR_Reset_Value и RCC_PLL_OD_Reset_Value.

6.109.2.7 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 FIN – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- $3,2 \text{ МГц} < \text{FIN} < 150 \text{ МГц}$,
- $800 \text{ КГц} < \text{FREF} < 8 \text{ МГц}$,
- $200 \text{ МГц} < \text{FVCO} < 500 \text{ МГц}$,

где частота фазового детектора FREF вычисляется по формуле:

$$\text{FREF} = \text{FIN} / (2 \times \text{NR}),$$

а частота FVCO вычисляется по формуле:

$$\text{FVCO} = \text{FIN} \times (\text{NF} / \text{NR})$$

Аргументы

RCC_PLLInit_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
--------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 256

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().

6.109.2.8 void RCC_PLLPowerDownCmd (FunctionalState State)

Управление режимом PowerDown PLL.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 307

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.109.2.9 void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.

Аргументы

RCC_PLLInit_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
--------------------	---

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 278

Перекрестные ссылки [RCC_PLLInit_TypeDef::RCC_PLLDiv](#), [RCC_PLLInit_TypeDef::RCC_PLLNF](#), [RCC_PLLInit_TypeDef::RCC_PLLNO](#), [RCC_PLLNO_Disable](#), [RCC_PLLInit_TypeDef::RCC_PLLNR](#), [RCC_PLLInit_TypeDef::RCC_PLLRef](#) и [RCC_PLLRef_XI_OSC](#).

6.109.2.10 `void RCC_SPIClkCmd (NT_SPI_TypeDef * SPIx, FunctionalState State)`

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 642

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#) и [IS_SPI_ALL_PERIPH](#).

6.109.2.11 `void RCC_SPIClkDivConfig (NT_SPI_TypeDef * SPIx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 604

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_RCC_CLK_DIV](#) и [IS_SPI_ALL_PERIPH](#).

6.109.2.12 `void RCC_SPIClkSel (NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)`

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 563

Перекрестные ссылки IS_RCC_SPI_CLK и IS_SPI_ALL_PERIPH.

6.109.2.13 void RCC_SysClkDiv2Out (FunctionalState State)

Включение генерации тактового сигнала с частой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState: <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 106

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_AltFunc_2, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_Out, GPIO_Init(), GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_0, GPIO_StructInit() и IS_FUNCTIONAL_STATE.

6.109.2.14 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_TypeDef .
------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 359

Перекрестные ссылки IS_RCC_SYS_CLK и RCC_WaitClkChange().

Используется в RCC_PLLAutoConfig().

6.109.2.15 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле `niietcm4_rcc.c` строка 388

6.109.2.16 `void RCC_UARTClkCmd (NT_UART_TypeDef * UARTx, FunctionalState State)`

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 520

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

6.109.2.17 `void RCC_UARTClkDivConfig (NT_UART_TypeDef * UARTx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 482

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_UART_ALL_PERIPH`.

6.109.2.18 `void RCC_UARTClkSel (NT_UART_TypeDef * UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)`

Настройка источника тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
RCC_UARTClk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 442

Перекрестные ссылки `IS_RCC_UART_CLK` и `IS_UART_ALL_PERIPH`.

6.109.2.19 `void RCC_USBClkCmd (FunctionalState State)`

Включение тактирования USB.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 419

Перекрестные ссылки `IS_FUNCTIONAL_STATE`.

6.109.2.20 `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 RCC_USBClk_TypeDef .
RCC_USBFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_rcc.c` строка 402

Перекрестные ссылки `IS_RCC_USB_CLK` и `IS_RCC_USB_FREQ`.

6.109.2.21 `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

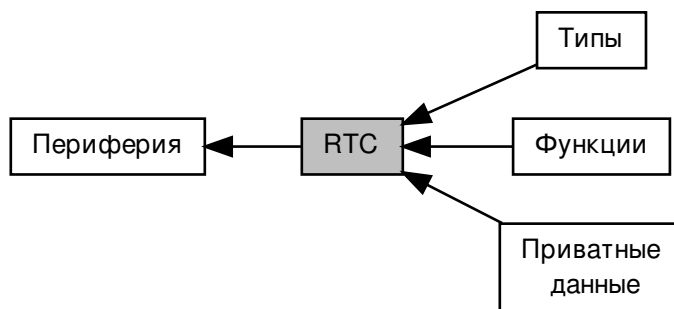
Перекрестные ссылки `RCC_CLK_CHANGE_TIMEOUT`.

Используется в `RCC_SysClkSel()`.

6.110 RTC

Драйвер для модуля часов реального времени.

Граф связей класса RTC:



Группы

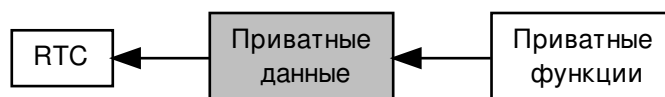
- [Типы](#)
- [Функции](#)
- [Приватные данные](#)

6.110.1 Подробное описание

Драйвер для модуля часов реального времени.

6.111 Приватные данные

Граф связей класса Приватные данные:



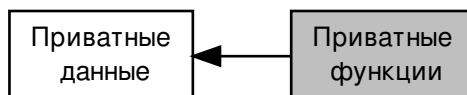
Группы

- [Приватные функции](#)

6.111.1 Подробное описание

6.112 Приватные функции

Граф связей класса Приватные функции:



Функции

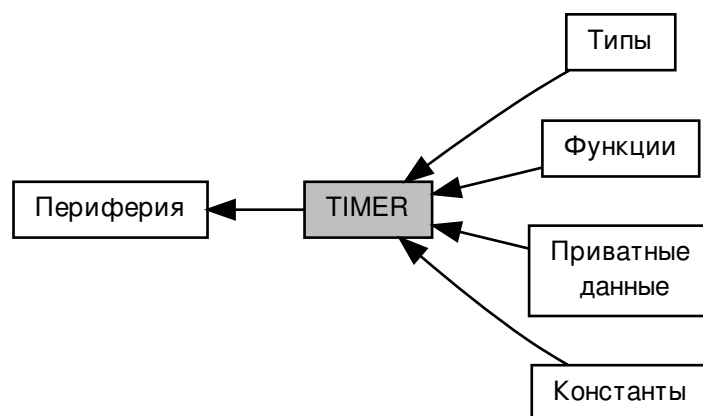
- `uint32_t bcd2hex (uint32_t a)`
- `uint32_t hex2bcd (uint32_t x)`
- `void RTC_ShadowUpd (FunctionalState State)`
- `void RTC_GetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)`
- `void RTC_GetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)`
- `void RTC_SetTime (RTC_Format_TypeDef RTC_Format, RTC_Time_TypeDef *RTC_Time)`
- `void RTC_SetDate (RTC_Format_TypeDef RTC_Format, RTC_Date_TypeDef *RTC_Date)`

6.112.1 Подробное описание

6.113 TIMER

Драйвер для таймеров

Граф связей класса TIMER:



Группы

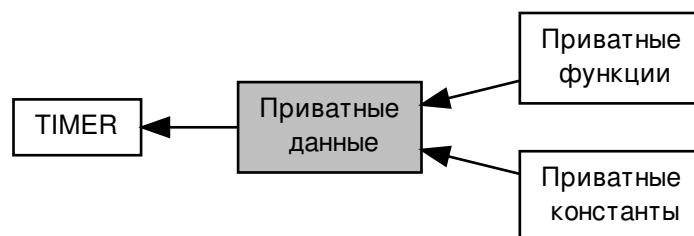
- [Типы](#)
- [Константы](#)
- [Функции](#)
- [Приватные данные](#)

6.113.1 Подробное описание

Драйвер для таймеров

6.114 Приватные данные

Граф связей класса Приватные данные:



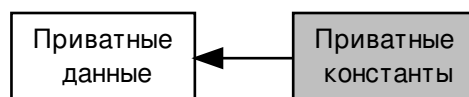
Группы

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

6.114.1 Подробное описание

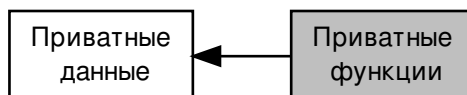
6.115 Приватные константы

Граф связей класса Приватные константы:



6.116 Приватные функции

Граф связей класса 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.

6.116.1 Подробное описание

6.116.2 Функции

6.116.2.1 void **TIMER_Cmd** (NT_TIMER_TypeDef * TIMERx, **FunctionalState** State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 65

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_TIMER_ALL_PERIPH`.

6.116.2.2 `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_ExtInput	Выбор режима работы. Параметр принимает любое значение из TIMER_ExtInput_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 171

Перекрестные ссылки `IS_TIMER_ALL_PERIPH`, `IS_TIMER_EXT_INPUT`, `TIMER_ExtInputCountClk` и `TIMER_ExtInput_CountEn`.

6.116.2.3 `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`.

6.116.2.4 `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.

6.116.2.5 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.

6.116.2.6 void TIMER_ITCmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

6.116.2.7 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.

6.116.2.8 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.

6.116.2.9 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.

6.116.2.10 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.

6.116.2.11 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

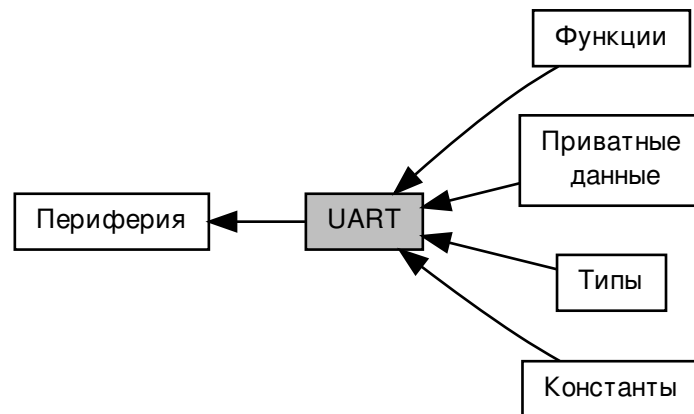
См. определение в файле niietcm4_timer.c строка 116

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

6.117 UART

Драйвер для приемопередатчиков UART.

Граф связей класса UART:



Группы

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

6.117.1 Подробное описание

Драйвер для приемопередатчиков UART.

Внимание

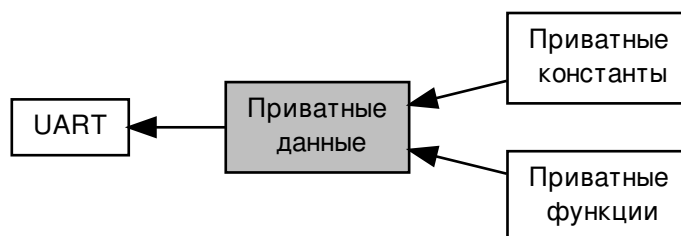
Драйвер позволяет управлять только внутренними настройками модулей UART. Соответствующие порты ввода-вывода, а также системное тактирование и сброс блоков необходимо настраивать отдельно.

[GPIO](#) : [Инициализация и деинициализация](#).

[RCC](#) : [Тактирование UART](#), [Управление сбросом](#).

6.118 Приватные данные

Граф связей класса Приватные данные:



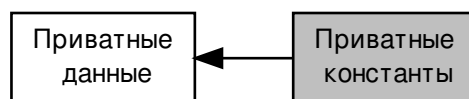
Группы

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

6.118.1 Подробное описание

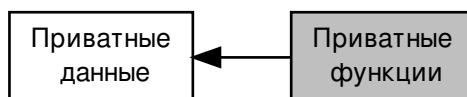
6.119 Приватные константы

Граф связей класса Приватные константы:



6.120 Приватные функции

Граф связей класса Приватные функции:



Функции

- void **UART_Cmd** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Разрешение работы выбранного UART.
- void **UART_BaudRateDivConfig** (NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t ←
t FracDiv)
Ручная настройка делителя для реализации необходимой скорости передачи.
- void **UART_Break** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Включение разрыва линии.
- void **UART_DeInit** (NT_UART_TypeDef *UARTx)
Устанавливает все регистры UART значениями по умолчанию.
- **OperationStatus** **UART_Init** (NT_UART_TypeDef *UARTx, **UART_Init_TypeDef** *UART ←
_InitStruct)
Инициализирует UARTx согласно параметрам структуры UART_InitStruct.
- void **UART_StructInit** (**UART_Init_TypeDef** *UART_InitStruct)
Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.
- void **UART_SendData** (NT_UART_TypeDef *UARTx, uint32_t Data)
Передача слова данных.
- uint32_t **UART_RecieveData** (NT_UART_TypeDef *UARTx)
Прием слова данных.
- **FlagStatus** **UART_FlagStatus** (NT_UART_TypeDef *UARTx, **UART_Flag_Typedef** UART ←
_Flag)
Запрос состояния выбранного флага.
- **FlagStatus** **UART_ErrorStatus** (NT_UART_TypeDef *UARTx, **UART_Error_Typedef** UAR ←
T_Error)
Запрос состояния выбранного флага ошибки.
- void **UART_ErrorStatusClear** (NT_UART_TypeDef *UARTx)
Очистка флагов ошибки.
- void **UART_ModemConfig** (NT_UART_TypeDef *UARTx, **UART_ModemInit_TypeDef** *U ←
ART_ModemInitStruct)
Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInit ←
Struct.
- void **UART_ModemStructInit** (**UART_ModemInit_TypeDef** *UART_ModemInitStruct)
Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.
- void **UART_ITFIFOLevelConfig** (NT_UART_TypeDef *UARTx, **UART_Dir_Typedef** UAR ←
T_Dir, **UART_FIFOLevel_TypeDef** UART_FIFOLevel)
Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генери-
роваться прерывание.

- void [UART_ITCmd](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_TypeDef](#) UART_ITSource, [FunctionalState](#) State)
Маскирование выбранных прерываний.
- [FlagStatus](#) [UART_ITRawStatus](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_TypeDef](#) UART_ITSource)
Запрос немаскированного состояния прерывания.
- [FlagStatus](#) [UART_ITMaskedStatus](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_TypeDef](#) UART_ITSource)
Запрос маскированного состояния прерывания.
- void [UART_ITStatusClear](#) (NT_UART_TypeDef *UARTx, [UART_ITSource_TypeDef](#) UART_ITSource)
Сброс флагов состояния выбранных прерываний.
- void [UART_DMABlkOnErrCmd](#) (NT_UART_TypeDef *UARTx, [FunctionalState](#) State)
Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.
- void [UART_DMACmd](#) (NT_UART_TypeDef *UARTx, [UART_Dir_TypeDef](#) UART_Dir, [FunctionalState](#) State)
Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

6.120.1 Подробное описание

6.120.2 Функции

6.120.2.1 void [UART_BaudRateDivConfig](#) (NT_UART_TypeDef * UARTx, uint32_t IntDiv, uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_uart.c` строка 88

Перекрестные ссылки `IS_UART_ALL_PERIPH`, `IS_UART_FRAC_DIV` и `IS_UART_INT_DIV`.

6.120.2.2 void [UART_Break](#) (NT_UART_TypeDef * UARTx, [FunctionalState](#) State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.120.2.3 void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.120.2.4 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().

6.120.2.5 void UART_DMABlkOnErrCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

6.120.2.6 void UART_DMACmd (NT_UART_TypeDef * UARTx, UART_Dir_TypeDef UART_Dir, FunctionalState State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DIR и UART_Dir_Rx.

6.120.2.7 FlagStatus UART_ErrorStatus (NT_UART_TypeDef * UARTx, UART_Error_TypeDef UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

6.120.2.8 void UART_ErrorStatusClear (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.120.2.9 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_TypeDef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_TypeDef .
-----------	---

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

6.120.2.10 `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 UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле niietcm4_uart.c строка 159

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DATA_WIDTH, IS_UART_FIFO_LEVEL, IS_UART_PARITY_BIT, IS_UART_STOP_BIT, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_Init_TypeDef::UART_FIFOEn, UART_Init_TypeDef::UART_FIFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Even, UART_ParityBit_High, UART_ParityBit_Low, UART_ParityBit_Odd, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit и UART_Init_TypeDef::UART_TxEn.

6.120.2.11 `void UART_ITCmd (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)`

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 410

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

6.120.2.12 `void UART_ITFIFOLevelConfig (NT_UART_TypeDef * UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)`

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор между буфером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFO← OLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_← TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и U←
ART_Dir_Rx.

6.120.2.13 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

6.120.2.14 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

6.120.2.15 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource↔_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

```
6.120.2.16 void UART_ModemConfig ( NT_UART_TypeDef * UARTx,
    UART_ModemInit_TypeDef * UART_ModemInitStruct )
```

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInit↔Struct.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_Modem↔Init_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit↔_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

```
6.120.2.17 void UART_ModemStructInit ( UART_ModemInit_TypeDef * UART_ModemInitStruct
    )
```

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef↔::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::↔UART_RTSEn.

```
6.120.2.18 uint32_t UART_RecieveData ( NT_UART_TypeDef * UARTx )
```

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

6.120.2.19 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

6.120.2.20 void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

Аргументы

UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

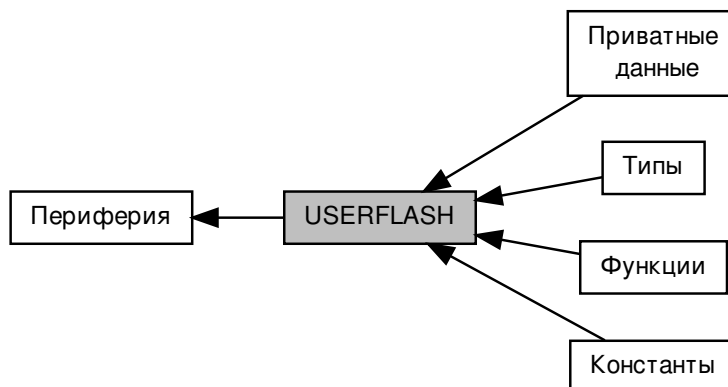
См. определение в файле niietcm4_uart.c строка 229

Перекрестные ссылки EXT_OSC_VALUE, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_DataWidth_8, UART_Init_TypeDef::UART_FIFOEn, UART_FIFOLevel_1_2, UART_Init_TypeDef::UART_FIFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Disable, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit, UART_StopBit_1 и UART_Init_TypeDef::UART_TxEn.

6.121 USERFLASH

Драйвер для пользовательской флеш-памяти.

Граф связей класса USERFLASH:



Группы

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

6.121.1 Подробное описание

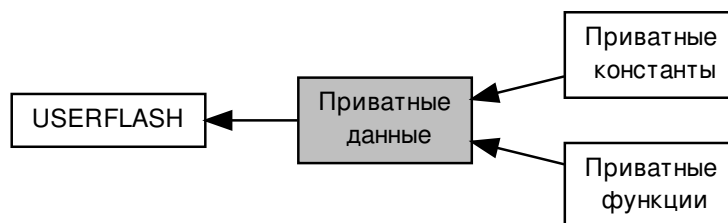
Драйвер для пользовательской флеш-памяти.

Внимание

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

6.122 Приватные данные

Граф связей класса Приватные данные:



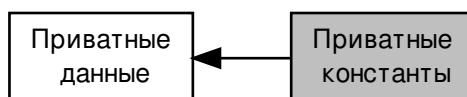
Группы

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

6.122.1 Подробное описание

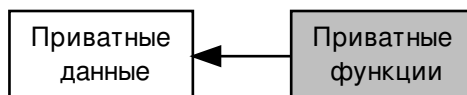
6.123 Приватные константы

Граф связей класса Приватные константы:



6.124 Приватные функции

Граф связей класса 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 флеш.
- void [USERFLASH_OperationStatusClear](#) ()
Очищает статус работы контроллера пользовательской флеш.
- void [USERFLASH_FullErase](#) ()
Полная очистка основной области пользовательской флеш.
- uint32_t [USERFLASH_Read](#) (uint32_t Address)
Чтение байта из основной области пользовательской флеш.
- void [USERFLASH_Write](#) (uint32_t Address, uint32_t Data)
Запись байта в основную область пользовательской флеш по указанному адресу.
- void [USERFLASH_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы основной области пользовательской флеш.
- uint32_t [USERFLASH_Info_Read](#) (uint32_t Address)
Чтение байта из информационной области пользовательской флеш.
- void [USERFLASH_Info_Write](#) (uint32_t Address, uint32_t Data)
Запись байта в информационную область пользовательской флеш по указанному адресу.
- void [USERFLASH_Info_PageErase](#) (uint32_t PageNum)
Стирание указанной страницы информационной области пользовательской флеш.
- void [USERFLASH_ITCmd](#) (FunctionalState State)
Включение прерывания по завершению чтения/записи/стирания.

6.124.1 Подробное описание

6.124.2 Функции

6.124.2.1 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.

6.124.2.2 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.

6.124.2.3 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.

6.124.2.4 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.

6.124.2.5 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

6.124.2.6 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

6.124.2.7 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

6.124.2.8 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().

6.124.2.9 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.

6.124.2.10 uint32_t USERFLASH_Read (uint32_t Address)

Чтение байта из основной области пользовательской флеш.

Аргументы

Address	Адрес чтения.
---------	---------------

Возвращаемые значения

Data	Байт данных.
------	--------------

См. определение в файле niietcm4_userflash.c строка 116

Перекрестные ссылки USERFLASH_MAGIC_KEY, USERFLASH_OPERATION_TIMEOUT, USERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

6.124.2.11 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

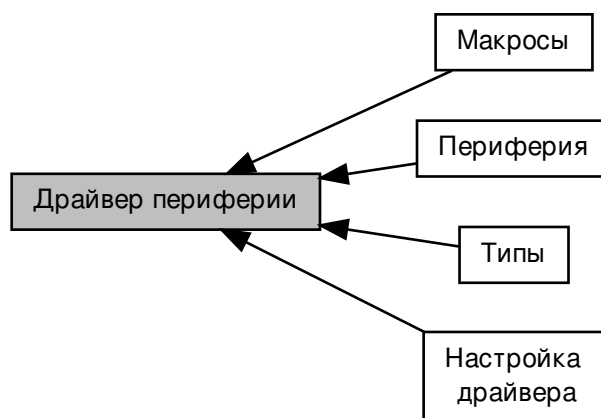
Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 148

Перекрестные ссылки USERFLASH_MAGIC_KEY.

6.125 Драйвер периферии

Граф связей класса Драйвер периферии:



Группы

- [Настройка драйвера](#)
- [Макросы](#)
- [Типы](#)
- [Периферия](#)

6.125.1 Подробное описание

6.126 Настройка драйвера

Граф связей класса 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.

6.126.1 Подробное описание

6.126.2 Макросы

6.126.2.1 `#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()`.

6.126.2.2 `#define INT_OSC_VALUE ((uint32_t)8000000)`

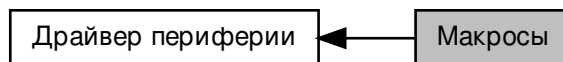
Определение частоты частоты внутреннего тактового генератора.

Конфигурируется автоматически в зависимости от выбранного целевого устройства. Частота внутреннего тактового генератора [Гц].

См. определение в файле `niietcm4.h` строка 88

6.127 Макросы

Граф связей класса Макросы:



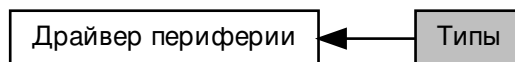
Макросы

- `#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.

6.127.1 Подробное описание

6.128 Типы

Граф связей класса 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`.

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.

6.128.1 Подробное описание

6.128.2 Макросы

6.128.2.1 `#define IS_GPIO_ALL_PERIPH(PERIPH)`

Макроопределение:

```

(((PERIPH) == NT_GPIOA) || \
 ((PERIPH) == NT_GPIOB) || \
 ((PERIPH) == NT_GPIOC) || \
 ((PERIPH) == NT_GPIOD) || \
 ((PERIPH) == NT_GPIOE) || \
 ((PERIPH) == NT_GPIOF) || \
 ((PERIPH) == NT_GPIOG) || \
 ((PERIPH) == NT_GPIOH))
  
```

Макрос проверки аргументов типа NT_GPIO_TypeDef.

См. определение в файле niietcm4.h строка 201

Используется в GPIO_ClearBits(), GPIO_DeInit(), GPIO_Init(), GPIO_ITCmd(), GPIO_ITConfig(), GPIO_ITStatusClear(), GPIO_QualCmd(), GPIO_QualConfig(), GPIO_Read(), GPIO_ReadBit(), GPIO_ReadMask(), GPIO_SetBits(), GPIO_SyncCmd(), GPIO_ToggleBits(), GPIO_Write(), GPIO_WriteBit() и GPIO_WriteMask().

6.128.2.2 #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().

6.128.2.3 #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().

6.128.2.4 #define IS_UART_ALL_PERIPH(PERIPH)

Макроопределение:

```
((PERIPH) == NT_UART0) || \
    ((PERIPH) == NT_UART1) || \
    ((PERIPH) == NT_UART2) || \
    ((PERIPH) == NT_UART3))
```

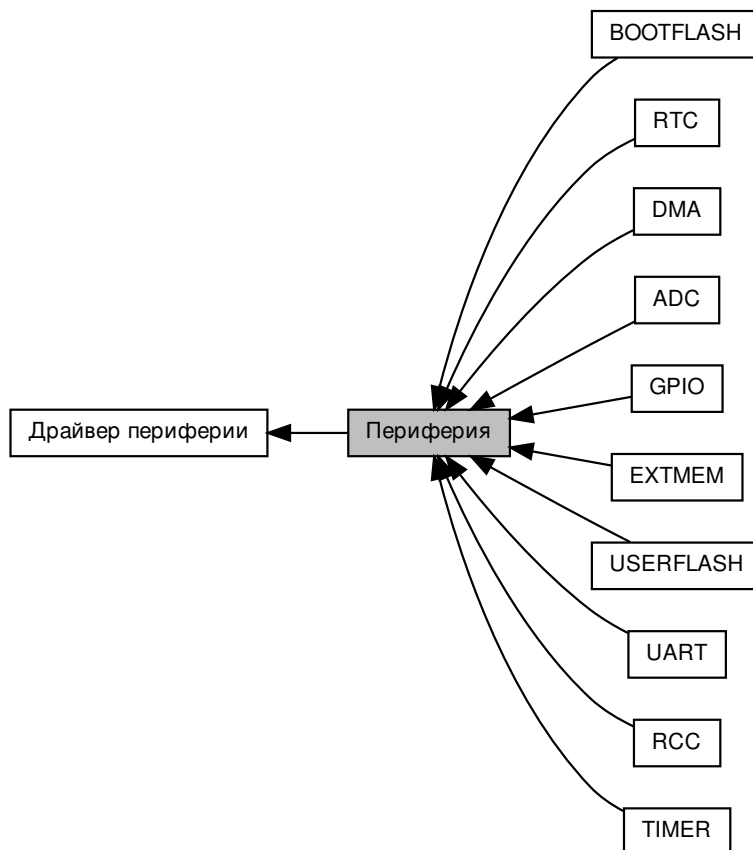
Макрос проверки аргументов типа NT_UART_TypeDef.

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_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().

6.129 Периферия

Граф связей класса Периферия:



Группы

- [ADC](#)
Драйвер для модулей АЦП, связанных с ними секвенсоров, а также цифровых компараторов.
- [BOOTFLASH](#)
Драйвер для загрузочной флеш-памяти.
- [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.

6.129.1 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:

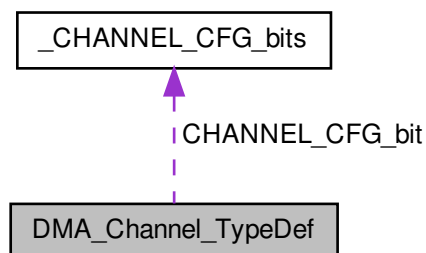
- [niietcm4_adc.h](#)

7.5 Структура DMA_Channel_TypeDef

Тип, описывающий структуру канала DMA.

```
#include <niietcm4_dma.h>
```

Граф связей класса DMA_Channel_TypeDef:



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

7.5.1 Подробное описание

Тип, описывающий структуру канала DMA.

См. определение в файле niietcm4_dma.h строка 228

7.5.2 Поля

7.5.2.1 uint32_t DMA_Channel_TypeDef::CHANNEL_CFG

Настройка канала

См. определение в файле niietcm4_dma.h строка 234

Используется в DMA_ChannelDeInit().

7.5.2.2 _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().

7.5.2.3 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().

7.5.2.4 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:

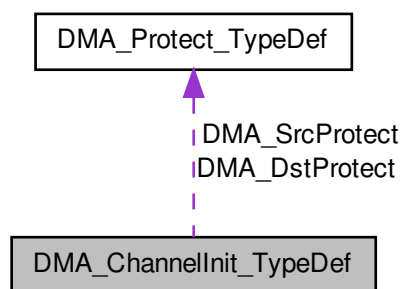
- [niietcm4_dma.h](#)

7.6 Структура DMA_ChannelInit_TypeDef

Структура инициализации канала DMA.

```
#include <niietcm4_dma.h>
```

Граф связей класса DMA_ChannelInit_TypeDef:



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

7.6.1 Подробное описание

Структура инициализации канала DMA.

См. определение в файле `niietcm4_dma.h` строка 383

7.6.2 Поля

7.6.2.1 `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()`.

7.6.2.2 `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()`.

7.6.2.3 `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()`.

7.6.2.4 `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()`.

7.6.2.5 `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()`.

7.6.2.6 `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()`.

7.6.2.7 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().

7.6.2.8 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().

7.6.2.9 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().

7.6.2.10 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().

7.6.2.11 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().

7.6.2.12 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:

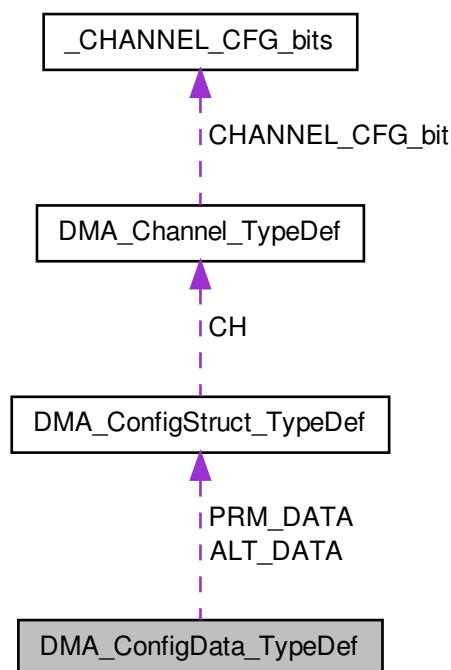
- [niietcm4_dma.h](#)

7.7 Структура DMA_ConfigData_TypeDef

Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ.

```
#include <niietcm4_dma.h>
```

Граф связей класса DMA_ConfigData_TypeDef:



Поля данных

- [DMA_ConfigStruct_TypeDef PRM_DATA](#)
- [uint32_t RESERVED0 \[32\]](#)
- [DMA_ConfigStruct_TypeDef ALT_DATA](#)
- [uint32_t RESERVED1 \[32\]](#)

7.7.1 Подробное описание

Совокупность из основной и управляющей структур DMA. Общий размер 1 кБ.

Внимание

Экземпляры этой структуры требуют обязательного выравнивания по 1024 байтам в адресном пространстве! Разрешенные адреса: 0xFFFFX000, 0xFFFFX400, 0xFFFFX800, 0xFFFFXC00.

См. определение в файле niietcm4_dma.h строка 256

7.7.2 Поля

7.7.2.1 DMA_ConfigStruct_TypeDef DMA_ConfigData_TypeDef::ALT_DATA

Альтернативная управляющая структура

См. определение в файле niietcm4_dma.h строка 260

7.7.2.2 DMA_ConfigStruct_TypeDef DMA_ConfigData_TypeDef::PRM_DATA

Основная управляющая структура

См. определение в файле niietcm4_dma.h строка 258

7.7.2.3 uint32_t DMA_ConfigData_TypeDef::RESERVED0[32]

Пустышка для неиспользованных 8 каналов DMA

См. определение в файле niietcm4_dma.h строка 259

7.7.2.4 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:

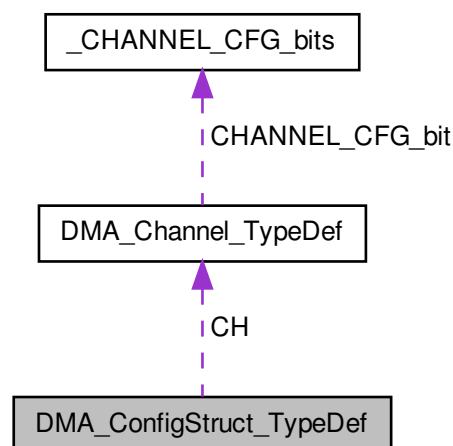
- [niietcm4_dma.h](#)

7.8 Структура DMA_ConfigStruct_TypeDef

Управляющая структура данных DMA.

```
#include <niietcm4_dma.h>
```

Граф связей класса DMA_ConfigStruct_TypeDef:



Поля данных

- [DMA_Channel_TypeDef CH](#) [24]

7.8.1 Подробное описание

Управляющая структура данных DMA.

См. определение в файле `niietcm4_dma.h` строка 244

7.8.2 Поля

7.8.2.1 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:

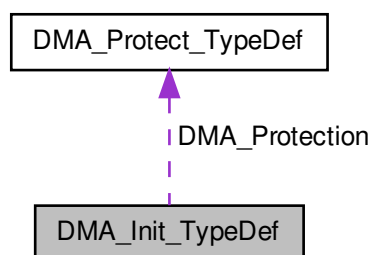
- [niietcm4_dma.h](#)

7.9 Структура DMA_Init_TypeDef

Структура инициализации контроллера DMA.

```
#include <niietcm4_dma.h>
```

Граф связей класса DMA_Init_TypeDef:



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`

7.9.1 Подробное описание

Структура инициализации контроллера DMA.

См. определение в файле `niietcm4_dma.h` строка 419

7.9.2 Поля

7.9.2.1 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()`.

7.9.2.2 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()`.

7.9.2.3 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()`.

7.9.2.4 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()`.

7.9.2.5 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()`.

7.9.2.6 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()`.

7.9.2.7 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](#)

7.10 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](#)

7.10.1 Подробное описание

Защита шины при чтении из источника или записи в приемник через DMA.

См. определение в файле niietcm4_dma.h строка 332

7.10.2 Поля

7.10.2.1 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().

7.10.2.2 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↔_StructInit().

7.10.2.3 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↔_StructInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_dma.h](#)

7.11 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`

7.11.1 Подробное описание

Структура инициализации внешней памяти.

См. определение в файле niietcm4_extmem.h строка 193

7.11.2 Поля

7.11.2.1 `uint32_t EXTMEM_Init_TypeDef::CEMask`

Маска адреса для генерации сигналов Cеп и Оеп. Параметр может принимать любое значение из диапазона 0-511.

См. определение в файле niietcm4_extmem.h строка 203

Используется в `EXTMEM_StructInit()`.

7.11.2.2 `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()`.

7.11.2.3 `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()`.

7.11.2.4 `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()`.

7.11.2.5 EXTMEM_WriteWaitState_TypeDef EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState ← State

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](#)

7.12 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`

7.12.1 Подробное описание

Структура инициализации GPIO.

См. определение в файле `niietcm4_gpio.h` строка 284

7.12.2 Поля

7.12.2.1 `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()`.

7.12.2.2 `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()`.

7.12.2.3 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()`.

7.12.2.4 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()`.

7.12.2.5 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()`.

7.12.2.6 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()`.

7.12.2.7 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()`.

7.12.2.8 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](#)

7.13 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](#)

7.13.1 Подробное описание

Структура инициализации PLL.

См. определение в файле niietcm4_rcc.h строка 375

7.13.2 Поля

7.13.2.1 [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 377

Используется в [RCC_PLLAutoConfig\(\)](#), [RCC_PLLInit\(\)](#) и [RCC_PLLStructInit\(\)](#).

7.13.2.2 [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 385

Используется в [RCC_PLLAutoConfig\(\)](#), [RCC_PLLInit\(\)](#) и [RCC_PLLStructInit\(\)](#).

7.13.2.3 [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 381

Используется в [RCC_PLLAutoConfig\(\)](#), [RCC_PLLInit\(\)](#) и [RCC_PLLStructInit\(\)](#).

7.13.2.4 [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 383

Используется в [RCC_PLLAutoConfig\(\)](#), [RCC_PLLInit\(\)](#) и [RCC_PLLStructInit\(\)](#).

7.13.2.5 [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 379

Используется в [RCC_PLLAutoConfig\(\)](#), [RCC_PLLInit\(\)](#) и [RCC_PLLStructInit\(\)](#).

Объявления и описания членов структуры находятся в файле:

- [niietcm4_rcc.h](#)

7.14 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`

7.14.1 Подробное описание

Структура даты.

См. определение в файле `niietcm4_rtc.h` строка 206

7.14.2 Поля

7.14.2.1 `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

7.14.2.2 `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

7.14.2.3 `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

7.14.2.4 `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](#)

7.15 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`

7.15.1 Подробное описание

Структура времени.

См. определение в файле `niietcm4_rtc.h` строка 178

7.15.2 Поля

7.15.2.1 `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

7.15.2.2 `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

7.15.2.3 `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

7.15.2.4 `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](#)

7.16 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

7.16.1 Подробное описание

Структура инициализации UART.

См. определение в файле niietcm4_uart.h строка 285

7.16.2 Поля

7.16.2.1 [uint32_t](#) UART_Init_TypeDef::UART_BaudRate

Желаемая скорость передачи данных в бит/с Максимальное значение 921600

См. определение в файле niietcm4_uart.h строка 294

Используется в [UART_Init\(\)](#) и [UART_StructInit\(\)](#).

7.16.2.2 [uint32_t](#) UART_Init_TypeDef::UART_ClkFreq

Значение текущей частоты в Гц, подведенной к блоку UART

См. определение в файле niietcm4_uart.h строка 293

Используется в [UART_Init\(\)](#) и [UART_StructInit\(\)](#).

7.16.2.3 [UART_DataWidth_TypeDef](#) UART_Init_TypeDef::UART_DataWidth

Количество передаваемых/принимаемых информационных бит. Параметр может принимать любое значение из [UART_DataWidth_TypeDef](#).

См. определение в файле niietcm4_uart.h строка 291

Используется в [UART_Init\(\)](#) и [UART_StructInit\(\)](#).

7.16.2.4 [FunctionalState](#) UART_Init_TypeDef::UART_FIFOEn

Разрешение режима FIFO буфера приемника и передатчика. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 300

Используется в `UART_Init()` и `UART_StructInit()`.

7.16.2.5 `UART_FIFOLevel_TypeDef` `UART_Init_TypeDef::UART_FIFOLevelRx`

Порог заполнения буфера приемника при срабатывании. Параметр может принимать любое значение из [UART_FIFOLevel_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 296

Используется в `UART_Init()` и `UART_StructInit()`.

7.16.2.6 `UART_FIFOLevel_TypeDef` `UART_Init_TypeDef::UART_FIFOLevelTx`

Порог заполнения буфера передатчика при срабатывании. Параметр может принимать любое значение из [UART_FIFOLevel_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 298

Используется в `UART_Init()` и `UART_StructInit()`.

7.16.2.7 `UART_ParityBit_TypeDef` `UART_Init_TypeDef::UART_ParityBit`

Выбор режима бита четности. Параметр может принимать любое значение из [UART_ParityBit_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 289

Используется в `UART_Init()` и `UART_StructInit()`.

7.16.2.8 `FunctionalState` `UART_Init_TypeDef::UART_RxEn`

Разрешение приема. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_uart.h` строка 302

Используется в `UART_Init()` и `UART_StructInit()`.

7.16.2.9 `UART_StopBit_TypeDef` `UART_Init_TypeDef::UART_StopBit`

Выбор режима передачи стопового бита. Параметр может принимать любое значение из [UART_StopBit_TypeDef](#).

См. определение в файле `niietcm4_uart.h` строка 287

Используется в `UART_Init()` и `UART_StructInit()`.

7.16.2.10 `FunctionalState` `UART_Init_TypeDef::UART_TxEn`

Разрешение передачи. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле `niietcm4_uart.h` строка 304

Используется в `UART_Init()` и `UART_StructInit()`.

Объявления и описания членов структуры находятся в файле:

- [niietcm4_uart.h](#)

7.17 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](#)

7.17.1 Подробное описание

Структура инициализации модемного режима.

См. определение в файле niietcm4_uart.h строка 269

7.17.2 Поля

7.17.2.1 FunctionalState UART_ModemInit_TypeDef::UART_CTSEn

Разрешение аппаратного управления потоком данных по линии CTS. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 277

Используется в UART_ModemConfig() и UART_ModemStructInit().

7.17.2.2 FunctionalState UART_ModemInit_TypeDef::UART_InvDTR

Управление инверсией сигнала на линии состояния модема DTR. Выключенная инверсия означает высокий уровень сигнала. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 271

Используется в UART_ModemConfig() и UART_ModemStructInit().

7.17.2.3 FunctionalState UART_ModemInit_TypeDef::UART_InvRTS

Управление инверсией сигнала на линии состояния модема RTS. Выключенная инверсия означает высокий уровень сигнала. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 273

Используется в UART_ModemConfig() и UART_ModemStructInit().

7.17.2.4 FunctionalState UART_ModemInit_TypeDef::UART_RTSEn

Разрешение аппаратного управления потоком данных по линии RTS. Параметр может принимать любое значение из [FunctionalState](#).

См. определение в файле niietcm4_uart.h строка 275

Используется в UART_ModemConfig() и UART_ModemStructInit().

Объявления и описания членов структуры находятся в файле:

- [niietcm4_uart.h](#)

Глава 8

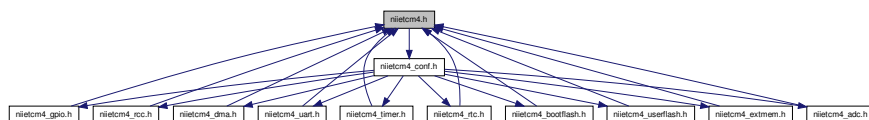
Файлы

8.1 Файл niietcm4.h

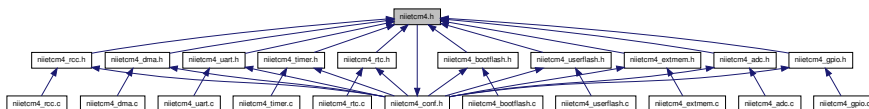
Это главный заголовочный файл драйвера, обычно включаемый в main.c.

```
#include "niietcm4_conf.h"
```

Граф включаемых заголовочных файлов для niietcm4.h:



Граф файлов, в которые включается этот файл:



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`.

Перечисления

- `enum FunctionalState { DISABLE = 0, ENABLE = 1 }`
Описывает логическое состояние периферии. Используется для операций включения/выключения периферийных блоков.
- `enum OperationStatus { OK = 0, ERROR = 1 }`
Описывает коды возврата для функций при выполнении какой-либо операции.
- `enum FlagStatus { Flag_CLEAR = 0, Flag_SET = 1 }`
Описывает возможные состояния флага при запросе его статуса.

8.1.1 Подробное описание

Это главный заголовочный файл драйвера, обычно включаемый в `main.c`.

Этот файл содержит:

- Главный заголовочный файл целевого устройства, с описанием всех регистров его периферии
- Область настройки драйвера, которая позволяет сконфигурировать:
 - Модель целевого устройства
 - Используемые тактовые частоты
- Макросы для доступа к регистрам периферии

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.1.2 Макросы

8.1.2.1 `#define EXT_OSC_VALUE ((uint32_t)12000000)`

Определение частоты используемого внешнего тактового генератора.

Совет: Чтобы избежать необходимости каждый раз изменять этот файл, можно передать определение устройства компилятору через ключ.

Например, для GCC ARM это выглядит так:

`-DEXT_OSC_VALUE=16000000`

Частота внешнего тактового генератора [Гц].

См. определение в файле niietcm4.h строка 73

Используется в `RCC_PLLAutoConfig()` и `UART_StructInit()`.

8.1.2.2 `#define INT_OSC_VALUE ((uint32_t)8000000)`

Определение частоты частоты внутреннего тактового генератора.

Конфигурируется автоматически в зависимости от выбранного целевого устройства. Частота внутреннего тактового генератора [Гц].

См. определение в файле niietcm4.h строка 88

8.1.2.3 `#define IS_GPIO_ALL_PERIPH(PERIPH)`

Макроопределение:

```
((PERIPH) == NT_GPIOA) || \
((PERIPH) == NT_GPIOB) || \
((PERIPH) == NT_GPIOC) || \
((PERIPH) == NT_GPIOD) || \
((PERIPH) == NT_GPIOE) || \
((PERIPH) == NT_GPIOF) || \
((PERIPH) == NT_GPIOG) || \
((PERIPH) == NT_GPIOH))
```

Макрос проверки аргументов типа `NT_GPIO_TypeDef`.

См. определение в файле niietcm4.h строка 201

Используется в `GPIO_ClearBits()`, `GPIO_DeInit()`, `GPIO_Init()`, `GPIO_ITCmd()`, `GPIO_ITConfig()`, `GPIO_ITStatusClear()`, `GPIO_QualCmd()`, `GPIO_QualConfig()`, `GPIO_Read()`, `GPIO_ReadBit()`, `GPIO_ReadMask()`, `GPIO_SetBits()`, `GPIO_SyncCmd()`, `GPIO_ToggleBits()`, `GPIO_Write()`, `GPIO_WriteBit()` и `GPIO_WriteMask()`.

8.1.2.4 `#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()`.

8.1.2.5 `#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()`.

8.1.2.6 `#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

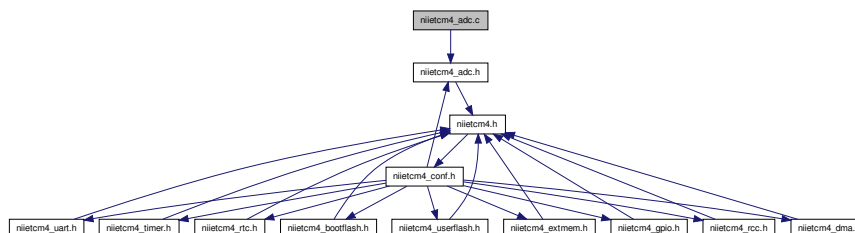
Используется в `RCC_UARTClkCmd()`, `RCC_UARTClkDivConfig()`, `RCC_UARTClkSel()`, `UART_BaudRateDivConfig()`, `UART_Break()`, `UART_Cmd()`, `UART_DeInit()`, `UART_DMABlkOnErrCmd()`, `UART_DMACmd()`, `UART_ErrorStatus()`, `UART_ErrorStatusClear()`, `UART_FlagStatus()`, `UART_Init()`, `UART_ITCmd()`, `UART_ITFIFOLevelConfig()`, `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"
```

Граф включаемых заголовочных файлов для niietcm4_adc.c:



Функции

- void **ADC_Cmd** (ADC_Module_TypeDef ADC_Module, FunctionalState State)
Включение модуля АЦП.
- void **ADC_DeInit** (ADC_Module_TypeDef ADC_Module)
Устанавливает все регистры модуля АЦП значениями по умолчанию.
- void **ADC_Init** (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *ADC_InitStruct)
Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.
- void **ADC_StructInit** (ADC_Init_TypeDef *ADC_InitStruct)
Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.
- void **ADC_DC_DeInit** (ADC_DC_Module_TypeDef ADC_DC_Module)
Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.
- void **ADC_DC_Init** (ADC_DC_Module_TypeDef ADC_DC_Module, ADC_DC_Init_TypeDef *ADC_DC_InitStruct)
Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры ADC_DC_InitStruct.
- void **ADC_DC_StructInit** (ADC_DC_Init_TypeDef *ADC_DC_InitStruct)
Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.
- void **ADC_SEQ_DeInit** (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Устанавливает все регистры выбранного секвенсора значениями по умолчанию.
- void **ADC_SEQ_Init** (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)
Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.
- void **ADC_SEQ_StructInit** (ADC_SEQ_Init_TypeDef *ADC_SEQ_InitStruct)
Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.
- void **ADC_SEQ_DMAConfig** (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)
Конфигурирует выбранный секвенсор для работы с DMA.
- void **ADC_SEQ_DMACmd** (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)
Включает для выбранного секвенсора генерирование запросов DMA.
- FlagStatus **ADC_SEQ_DMAErrorStatus** (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Проверка статуса ошибки, когда при наличии двух обрабатываемых запросов DMA от выбранного секвенсора, пришел третий запрос, который не может быть обработан.
- void **ADC_SEQ_DMAErrorStatusClear** (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)
Сброс статуса ошибки DMA.
- void **ADC_DC_ITGenCmd** (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)
Разрешает компаратору генерировать сигнал прерывания.

- void [ADC_DC_ITMaskCmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
 Маскирование сигнала прерывания цифрового компаратора.
- void [ADC_DC_ITCmd](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [FunctionalState](#) State)
 Включение прерывания компаратора и одновременное маскирование сигнала этого прерывания. При этом, эти же действия можно выполнить путем ручного вызова соответствующих функций: [ADC_DC_ITGenCmd](#) и [ADC_DC_ITMaskCmd](#).
- void [ADC_DC_ITConfig](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module, [ADC_DC_Mode_TypeDef](#) ADC_DC_Mode, [ADC_DC_Condition_TypeDef](#) ADC_DC_Condition)
 Настройка условия вызова прерывания цифрового компаратора. Условия вызова прерывания и условия срабатывания выходного триггера компаратора могут не совпадать.
- [FlagStatus](#) [ADC_DC_ITRawStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
 Проверка флагов немаскированных прерываний.
- [FlagStatus](#) [ADC_DC_ITMaskedStatus](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
 Проверка флагов маскированных прерываний.
- void [ADC_DC_ITStatusClear](#) ([ADC_DC_Module_TypeDef](#) ADC_DC_Module)
 Общий сброс флагов прерывания цифрового компаратора. Сбрасывает как маскированные, так и немаскированные флаги.
- void [ADC_SEQ_ITCmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)
 Включение прерывания секвенсора.
- void [ADC_SEQ_ITConfig](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, uint32_t ADC_SEQ_ITRate, [FunctionalState](#) ADC_SEQ_ITCountSEQRst)
 Настройка вызова прерывания секвенсора.
- uint32_t [ADC_SEQ_GetITCount](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Текущее значение счетчика измерений, который используется для генерации прерывания секвенсора.
- void [ADC_SEQ_ITCountRst](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Сброс счетчика прерываний секвенсора.
- [FlagStatus](#) [ADC_SEQ_ITRawStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Проверка флагов немаскированных прерываний.
- [FlagStatus](#) [ADC_SEQ_ITMaskedStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Проверка флагов маскированных прерываний.
- void [ADC_SEQ_ITStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Общий сброс флагов прерывания секвенсора. Сбрасывает как маскированные, так и немаскированные флаги.
- void [ADC_SEQ_Cmd](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module, [FunctionalState](#) State)
 Включение секвенсора.
- void [ADC_SEQ_SWReq](#) ()
 Программный запуск измерений всех разрешенных секвенсоров.
- uint32_t [ADC_SEQ_GetFIFOData](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Получение результата измерений из буфера секвенсора.
- uint32_t [ADC_SEQ_GetConversionCount](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Получение количества измерений, проведенных модулями АЦП с момента запуска секвенсора.
- uint32_t [ADC_SEQ_GetFIFOLoad](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Получение количества измерений, сохраненных в буфере секвенсора.
- [FlagStatus](#) [ADC_SEQ_FIFOFullStatus](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)
 Проверка флага заполнения буфера секвенсора. Если флаг установлен, то значит что буфер заполнен и все последующие записи в буфер будут блокироваться до появления как минимум одной свободной ячейки.
- void [ADC_SEQ_FIFOFullStatusClear](#) ([ADC_SEQ_Module_TypeDef](#) ADC_SEQ_Module)

- Сброс флага заполнения буфера секвенсора.
- `FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Проверка флага пустоты буфера секвенсора. Флаг установлен когда буфер полностью пуст.
- `void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)`
Сброс флага пустоты буфера секвенсора.
- `void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)`
Включение выходного триггера цифрового компаратора.
- `uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)`
Значение результата измерения, которое последним использовалось компаратором при проверке на соответствие условиям.
- `FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)`
Проверка состояния выходного триггера компаратора.
- `void ADC_DC_TrigStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)`
Сброс выходного триггера цифрового компаратора.

8.2.1 Подробное описание

Файл содержит реализацию всех функций для работы с модулями АЦП, секвенсорами, цифровыми компараторами.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

10.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.2.2 Функции

8.2.2.1 `void ADC_Cmd (ADC_Module_TypeDef ADC_Module, FunctionalState State)`

Включение модуля АЦП.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 115

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1024

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.3 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 307

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.4 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1040

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.5 void ADC_DC_Init (ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Инициализирует выбранный модуль цифрового компаратора согласно параметрам структуры ADC_DC_InitStruct.

Аргументы

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

См. определение в файле niietcm4_adc.c строка 326

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Init_TypeDef::ADC_DC_Mode, ADC_DC_Init_TypeDef::ADC_DC_ThresholdHigh, ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow, IS_ADC_DC, IS_ADC_DC_CHANNEL, IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_THRESHOLD.

8.2.2.6 void ADC_DC_ITCmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 596

Перекрестные ссылки ADC_DC_ITGenCmd(), ADC_DC_ITMaskCmd(), IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.7 void ADC_DC_ITConfig (ADC_DC_Module_TypeDef ADC_DC_Module,
ADC_DC_Mode_TypeDef ADC_DC_Mode, ADC_DC_Condition_TypeDef
ADC_DC_Condition)

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 620

Перекрестные ссылки IS_ADC_DC_CONDITION, IS_ADC_DC_MODE и IS_ADC_DC_MODULE.

```
8.2.2.8 void ADC_DC_ITGenCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                             FunctionalState State )
```

Разрешает компаратору генерировать сигнал прерывания.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 553

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

```
8.2.2.9 void ADC_DC_ITMaskCmd ( ADC_DC_Module_TypeDef ADC_DC_Module,
                               FunctionalState State )
```

Маскирование сигнала прерывания цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 570

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

Используется в ADC_DC_ITCmd().

```
8.2.2.10 FlagStatus ADC_DC_ITMaskedStatus ( ADC_DC_Module_TypeDef ADC_DC_Module
                                             )
```

Проверка флагов маскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 662

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.11 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 637

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.12 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 688

Перекрестные ссылки IS_ADC_DC_MODULE.

8.2.2.13 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 349

Перекрестные ссылки ADC_DC_Init_TypeDef::ADC_DC_Channel, ADC_DC_Channel_None, ADC_DC_Init_TypeDef::ADC_DC_Condition, ADC_DC_Condition_Low, ADC_DC_Init_↔

_TypeDef::ADC_DC_Mode, ADC_DC_Mode_Single, ADC_DC_Init_TypeDef::ADC_DC_↔ThresholdHigh и ADC_DC_Init_TypeDef::ADC_DC_ThresholdLow.

8.2.2.14 FlagStatus ADC_DC_TrigStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка состояния выходного триггера компаратора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1058

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 ADC_↔C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 1086

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

8.2.2.16 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

ADC_Module	Выбор модуля АЦП. Параметр принимает любое значение из ADC_Module_↔TypeDef .
------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 130

Перекрестные ссылки ADC_Module_0, ADC_Module_1, ADC_Module_10, ADC_Module_11, A↔DC_Module_2, ADC_Module_3, ADC_Module_4, ADC_Module_5, ADC_Module_6, ADC_↔Module_7, ADC_Module_8, ADC_Module_9 и IS_ADC_MODULE.

```
8.2.2.17 void ADC_Init ( ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef *  
ADC_InitStruct )
```

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

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

См. определение в файле niietcm4_adc.c строка 206

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

```
8.2.2.18 void ADC_SEQ_Cmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                          FunctionalState State )
```

Включение секвенсора.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 855

Перекрестные ссылки [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

```
8.2.2.19 void ADC_SEQ_DeInit ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module )
```

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

ADC_SEQ_Module	Выбор модуля цифрового компаратора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 365

Перекрестные ссылки [ADC_SEQ_Module_0](#), [ADC_SEQ_Module_1](#), [ADC_SEQ_Module_2](#), [ADC_SEQ_Module_3](#), [ADC_SEQ_Module_4](#), [ADC_SEQ_Module_5](#), [ADC_SEQ_Module_6](#), [ADC_SEQ_Module_7](#) и [IS_ADC_SEQ_MODULE](#).

```
8.2.2.20 void ADC_SEQ_DMAMCmd ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
                              FunctionalState State )
```

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 496

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.2.2.21 void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 479

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

8.2.2.22 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 512

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.23 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 537

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.24 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 983

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.25 void ADC_SEQ_FIFOEmptyStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага пустоты буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1008

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.26 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 943

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.27 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 968

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.28 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 905

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.29 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Получение результата измерений из буфера секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 887

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
8.2.2.30 uint32_t ADC_SEQ_GetFIFOLoad ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)
```

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 923

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.31 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 757

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.32 void ADC_SEQ_Init (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 425

Перекрестные ссылки ADC_SEQ_Init_TypeDef::ADC_Channels, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount, ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay, ADC_SEQ_Init_TypeDef::ADC_SEQ_DC, ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent, ADC_SEQ_Init_TypeDef::ADC_SEQ_SWReqEn, IS_ADC_CHANNEL, IS_ADC_DC, IS_ADC_SEQ_CONVERSION_COUNT, IS_ADC_SEQ_CONVERSION_DELAY, IS_ADC_SEQ_MODULE, IS_ADC_SEQ_START_EVENT и IS_FUNCTIONAL_STATE.

8.2.2.33 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
FunctionalState State)

Включение прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 704

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

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

Настройка вызова прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 732

Перекрестные ссылки IS_ADC_SEQ_IT_RATE, IS_ADC_SEQ_MODULE и IS_FUNCTIONAL↔
L_STATE.

8.2.2.35 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 775

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.36 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 814

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.37 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 789

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.38 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 839

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.2.2.39 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 460

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 875

8.2.2.41 void ADC_StructInit (ADC_Init_TypeDef * ADC_InitStruct)

Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 290

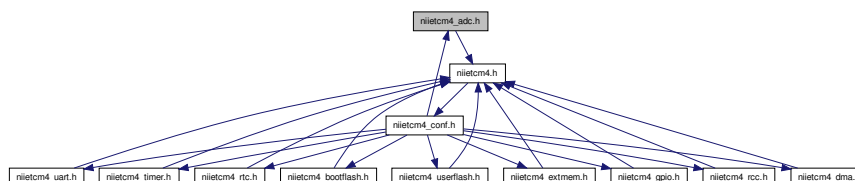
Перекрестные ссылки ADC_Init_TypeDef::ADC_Average, ADC_Average_Disable, ADC_Init_↔TypeDef::ADC_Measure_A, ADC_Init_TypeDef::ADC_Measure_B, ADC_Measure_Single, ADC↔_Init_TypeDef::ADC_Mode, ADC_Mode_Powerdown, ADC_Init_TypeDef::ADC_Phase, ADC_↔Init_TypeDef::ADC_Resolution и ADC_Resolution_12bit.

8.3 Файл niietcm4_adc.h

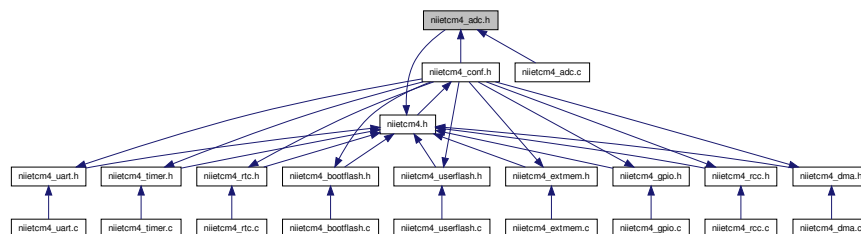
Файл содержит все прототипы функций для работы с АЦП, секвенсорами, цифровыми компараторами.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_adc.h:



Граф файлов, в которые включается этот файл:



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)0x00FFFFFF)`
- `#define IS_ADC_CHANNEL(CHANNEL) (((CHANNEL) & (uint32_t)0xFF000000) == ((uint32_t)0x00000000))`
Макрос проверки попадания масок каналов в допустимый диапазон.
- `#define ADC_DC_None ((uint32_t)0x00000000)`

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

Макрос проверки попадания масок компараторов в допустимый диапазон.

- `#define ADC_SEQ_0 ((uint32_t)0x00000001)`
- `#define ADC_SEQ_1 ((uint32_t)0x00000002)`
- `#define ADC_SEQ_2 ((uint32_t)0x00000004)`
- `#define ADC_SEQ_3 ((uint32_t)0x00000008)`
- `#define ADC_SEQ_4 ((uint32_t)0x00000010)`
- `#define ADC_SEQ_5 ((uint32_t)0x00000020)`
- `#define ADC_SEQ_6 ((uint32_t)0x00000040)`
- `#define ADC_SEQ_7 ((uint32_t)0x00000080)`
- `#define IS_ADC_SEQ(SEQ) (((SEQ) != (uint32_t)0x00000000) && (((SEQ) & (uint32_t)0x←FFFFFFF00) == ((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) (((CONVERSIO←N_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) ((CONVERSIO←N_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@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"

8.3.2 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 ADC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 115

Перекрестные ссылки IS_ADC_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.2 void ADC_DC_Cmd (ADC_DC_Module_TypeDef ADC_DC_Module, FunctionalState State)

Включение выходного триггера цифрового компаратора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1024

Перекрестные ссылки IS_ADC_DC_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.3 void ADC_DC_DeInit (ADC_DC_Module_TypeDef ADC_DC_Module)

Устанавливает все регистры выбранного цифрового компаратора значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 307

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.4 uint32_t ADC_DC_GetLastData (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1040

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

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

См. определение в файле niietcm4_adc.c строка 326

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

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 596

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

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 620

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 553

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 ADC_↔ C_DC_Module_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 570

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 ADC_↔ C_DC_Module_TypeDef .
--------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 662

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.11 FlagStatus ADC_DC_ITRawStatus (ADC_DC_Module_TypeDef ADC_DC_Module)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 637

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.12 void ADC_DC_ITStatusClear (ADC_DC_Module_TypeDef ADC_DC_Module)

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

Аргументы

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

Возвращаемые значения

нет	
-----	--

См. определение в файле niietcm4_adc.c строка 688

Перекрестные ссылки IS_ADC_DC_MODULE.

8.3.4.13 void ADC_DC_StructInit (ADC_DC_Init_TypeDef * ADC_DC_InitStruct)

Заполнение каждого члена структуры ADC_DC_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_adc.c строка 349

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 ADC_DC_↔ Module_TypeDef .
--------------------	---

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 1058

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 AD_↔ C_DC_Module_TypeDef .
--------------------	--

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1086

Перекрестные ссылки ADC_DC_Module_0, ADC_DC_Module_1 и IS_ADC_DC_MODULE.

8.3.4.16 void ADC_DeInit (ADC_Module_TypeDef ADC_Module)

Устанавливает все регистры модуля АЦП значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 130

Перекрестные ссылки ADC_Module_0, ADC_Module_1, ADC_Module_10, ADC_Module_11, ADC_Module_2, ADC_Module_3, ADC_Module_4, ADC_Module_5, ADC_Module_6, ADC_Module_7, ADC_Module_8, ADC_Module_9 и IS_ADC_MODULE.

8.3.4.17 void ADC_Init (ADC_Module_TypeDef ADC_Module, ADC_Init_TypeDef * ADC_InitStruct)

Инициализирует выбранный модуль АЦП согласно параметрам структуры ADC_InitStruct.

Аргументы

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

См. определение в файле niietcm4_adc.c строка 206

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

8.3.4.18 void ADC_SEQ_Cmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение секвенсора.

Аргументы

ADC_SEQ_Module	Выбор секвенсора. Параметр принимает любое значение из ADC_SEQ_Module_TypeDef .
----------------	---

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 855

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.19 void ADC_SEQ_DeInit (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Устанавливает все регистры выбранного секвенсора значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 365

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

8.3.4.20 void ADC_SEQ_DMAMCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
FunctionalState State)

Включает для выбранного секвенсора генерирование запросов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 496

Перекрестные ссылки IS_ADC_SEQ_MODULE и IS_FUNCTIONAL_STATE.

8.3.4.21 void ADC_SEQ_DMAConfig (ADC_SEQ_Module_TypeDef ADC_SEQ_Module,
ADC_SEQ_FIFOLevel_TypeDef ADC_SEQ_FIFOLevel)

Конфигурирует выбранный секвенсор для работы с DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 479

Перекрестные ссылки IS_ADC_SEQ_FIFO_LEVEL и IS_ADC_SEQ_MODULE.

8.3.4.22 FlagStatus ADC_SEQ_DMAErrorStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 512

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.23 void ADC_SEQ_DMAErrorStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс статуса ошибки DMA.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 537

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.24 FlagStatus ADC_SEQ_FIFOEmptyStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 983

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
8.3.4.25 void ADC_SEQ_FIFOEmptyStatusClear ( ADC_SEQ_Module_TypeDef  
        ADC_SEQ_Module )
```

Сброс флага пустоты буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 1008

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.26 FlagStatus ADC_SEQ_FIFOFullStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 943

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.27 void ADC_SEQ_FIFOFullStatusClear (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Сброс флага заполнения буфера секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 968

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.28 uint32_t ADC_SEQ_GetConversionCount (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 905

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.29 uint32_t ADC_SEQ_GetFIFOData (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Получение результата измерений из буфера секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 887

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.30 uint32_t ADC_SEQ_GetFIFOLoad (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Получение количества измерений, сохраненных в буфере секвенсора.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 923

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.31 uint32_t ADC_SEQ_GetITCount (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 757

Перекрестные ссылки IS_ADC_SEQ_MODULE.

```
8.3.4.32 void ADC_SEQ_Init ( ADC_SEQ_Module_TypeDef ADC_SEQ_Module,  
ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct )
```

Инициализирует выбранный секвенсор согласно параметрам структуры ADC_SEQ_InitStruct.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 425

Перекрестные ссылки [ADC_SEQ_Init_TypeDef::ADC_Channels](#), [ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionCount](#), [ADC_SEQ_Init_TypeDef::ADC_SEQ_ConversionDelay](#), [ADC_SEQ_Init_TypeDef::ADC_SEQ_DC](#), [ADC_SEQ_Init_TypeDef::ADC_SEQ_StartEvent](#), [ADC_SEQ_Init_TypeDef::ADC_SEQ_SWReqEn](#), [IS_ADC_CHANNEL](#), [IS_ADC_DC](#), [IS_ADC_SEQ_CONVERSION_COUNT](#), [IS_ADC_SEQ_CONVERSION_DELAY](#), [IS_ADC_SEQ_MODULE](#), [IS_ADC_SEQ_START_EVENT](#) и [IS_FUNCTIONAL_STATE](#).

8.3.4.33 void ADC_SEQ_ITCmd (ADC_SEQ_Module_TypeDef ADC_SEQ_Module, FunctionalState State)

Включение прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 704

Перекрестные ссылки [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

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

Настройка вызова прерывания секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 732

Перекрестные ссылки [IS_ADC_SEQ_IT_RATE](#), [IS_ADC_SEQ_MODULE](#) и [IS_FUNCTIONAL_STATE](#).

8.3.4.35 void ADC_SEQ_ITCountRst (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

Сброс счетчика прерываний секвенсора.

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 775

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.36 FlagStatus ADC_SEQ_ITMaskedStatus (ADC_SEQ_Module_TypeDef
ADC_SEQ_Module)

Проверка флагов маскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 814

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.37 FlagStatus ADC_SEQ_ITRawStatus (ADC_SEQ_Module_TypeDef ADC_SEQ_Module
)

Проверка флагов немаскированных прерываний.

Аргументы

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

Возвращаемые значения

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

См. определение в файле niietcm4_adc.c строка 789

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.38 void ADC_SEQ_ITStatusClear (ADC_SEQ_Module_TypeDef ADC_SEQ_Module)

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

Аргументы

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

Возвращаемые значения

нет

См. определение в файле niietcm4_adc.c строка 839

Перекрестные ссылки IS_ADC_SEQ_MODULE.

8.3.4.39 void ADC_SEQ_StructInit (ADC_SEQ_Init_TypeDef * ADC_SEQ_InitStruct)

Заполнение каждого члена структуры ADC_SEQ_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 460

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 875

8.3.4.41 void ADC_StructInit (ADC_Init_TypeDef * ADC_InitStruct)

Заполнение каждого члена структуры ADC_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_adc.c строка 290

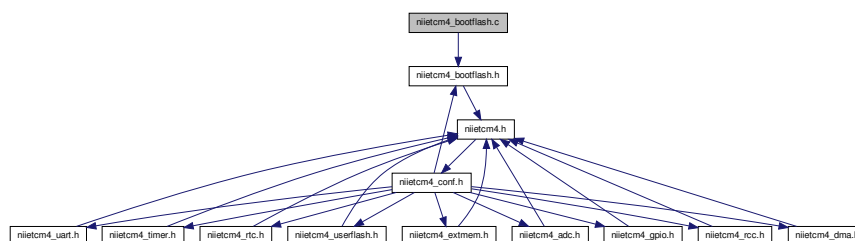
Перекрестные ссылки ADC_Init_TypeDef::ADC_Average, ADC_Average_Disable, ADC_Init_↔TypeDef::ADC_Measure_A, ADC_Init_TypeDef::ADC_Measure_B, ADC_Measure_Single, ADC↔_Init_TypeDef::ADC_Mode, ADC_Mode_Powerdown, ADC_Init_TypeDef::ADC_Phase, ADC_↔Init_TypeDef::ADC_Resolution и ADC_Resolution_12bit.

8.4 Файл niietcm4_bootflash.c

Файл содержит реализацию всех функции для работы с загрузочной флеш.

```
#include "niietcm4_bootflash.h"
```

Граф включаемых заголовочных файлов для niietcm4_bootflash.c:



Функции

- void **BOOTFLASH_Init** (uint32_t SysClkFreq)
Инициализирует тайминги доступа для контроллера загрузочной флеш.
- **BOOTFLASH_Status_TypeDef** **BOOTFLASH_OperationStatus** ()
Статус работы контроллера загрузочной флеш.
- void **BOOTFLASH_OperationStatusClear** ()
Очищает статус работы контроллера загрузочной флеш.
- void **BOOTFLASH_FullErase** ()
Полная очистка основной области загрузочной флеш.
- void **BOOTFLASH_Write** (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в основную область загрузочной флеш, начиная с указанного адреса.
- void **BOOTFLASH_PageErase** (uint32_t PageNum)
Стирание указанной страницы основной области загрузочной флеш.
- void **BOOTFLASH_Info_Write** (uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t Data2, uint32_t Data3)
Запись 128 бит информации в информационную область загрузочной флеш, начиная с указанного адреса.
- void **BOOTFLASH_Info_PageErase** (uint32_t PageNum)
Стирание указанной страницы информационной области загрузочной флеш.
- void **BOOTFLASH_ITCmd** (**FunctionalState** State)
Включение прерывания по завершению чтения/записи/стирания.

8.4.1 Подробное описание

Файл содержит реализацию всех функции для работы с загрузочной флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

07.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С 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"

8.4.2 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 флэш.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 75

8.4.2.5 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.4.2.6 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_bootflash.c строка 88

8.4.2.7 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 102

8.4.2.8 void BOOTFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области загрузочной 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.

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 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

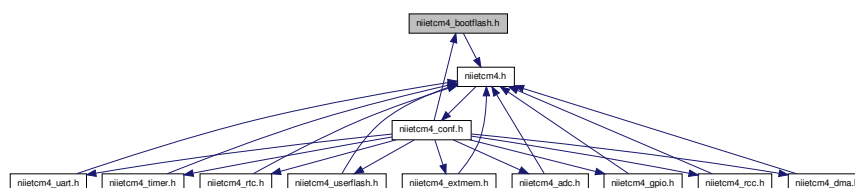
Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.5 Файл niietcm4_bootflash.h

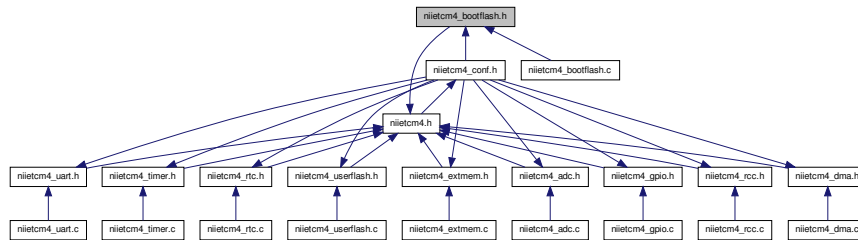
Файл содержит все прототипы функций для загрузочной флеш.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_bootflash.h:



Граф файлов, в которые включается этот файл:



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_INFO_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

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↵
S*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*BOOTS↵
FLASH_PAGE_TOTAL)`

Общий размер основной области.

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

8.5.2.7 `#define IS_BOOTFLASH_STATUS(STATUS)`

Макроопределение:

```
((STATUS) == BOOTFLASH\_Status\_None) || \
((STATUS) == BOOTFLASH\_Status\_Complete) || \
((STATUS) == BOOTFLASH\_Status\_Error)
```

Макрос проверки аргументов типа [BOOTFLASH_Status_TypeDef](#).

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

8.5.3 Перечисления

8.5.3.1 `enum BOOTFLASH_Status_TypeDef`

Статус работы контроллера загрузочной флеш-памяти.

Элементы перечислений

`BOOTFLASH_Status_None` Операция выполняется или отсутствует.

`BOOTFLASH_Status_Complete` Операция успешно завершена.

`BOOTFLASH_Status_Error` Операция завершена с ошибкой.

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

8.5.4 Функции

8.5.4.1 void BOOTFLASH_FullErase ()

Полная очистка основной области загрузочной флеш.

Возвращаемые значения

Нет.	
------	--

См. определение в файле niietcm4_bootflash.c строка 112

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.5.4.2 void BOOTFLASH_Info_PageErase (uint32_t PageNum)

Стирание указанной страницы информационной области загрузочной флеш.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 179

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_↔ BOOTFLASH_INFO_PAGE_NUM.

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

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

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_bootflash.c строка 163

Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.5.4.4 void BOOTFLASH_Init (uint32_t SysClkFreq)

Инициализирует тайминги доступа для контроллера загрузочной флеш.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 75

8.5.4.5 void BOOTFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 194

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.5.4.6 BOOTFLASH_Status_TypeDef BOOTFLASH_OperationStatus ()

Статус работы контроллера загрузочной флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_bootflash.c строка 88

8.5.4.7 void BOOTFLASH_OperationStatusClear ()

Очищает статус работы контроллера загрузочной флэш.

Возвращаемые значения

Нет.

См. определение в файле niietcm4_bootflash.c строка 102

8.5.4.8 void BOOTFLASH_PageErase (uint32_t PageNum)

Стирание указанной страницы основной области загрузочной флэш.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_bootflash.c строка 144

Перекрестные ссылки BOOTFLASH_MAGIC_KEY, BOOTFLASH_PAGE_SIZE_BYTES и IS_BOOTFLASH_PAGE_NUM.

```
8.5.4.9 void BOOTFLASH_Write ( uint32_t Address, uint32_t Data0, uint32_t Data1, uint32_t  
Data2, uint32_t Data3 )
```

Запись 128 бит информации в основную область загрузочной 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

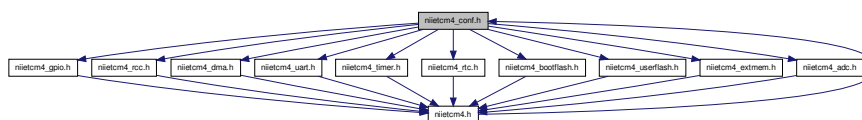
Перекрестные ссылки BOOTFLASH_MAGIC_KEY.

8.6 Файл niietcm4_conf.h

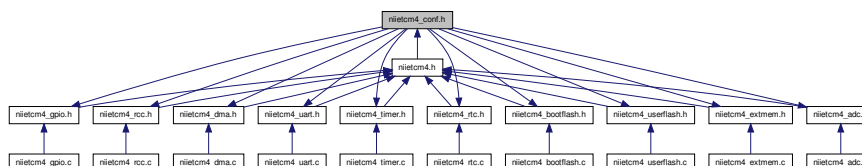
Файл конфигурации драйвера.

```
#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"
```

Граф включаемых заголовочных файлов для niietcm4_conf.h:



Граф файлов, в которые включается этот файл:



Макросы

- `#define assert_param(expr) ((void)0)`

8.6.1 Подробное описание

Файл конфигурации драйвера.

Основные функции:

- Исключение исходного кода для неиспользуемой периферии.
- Включение assert'ов.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

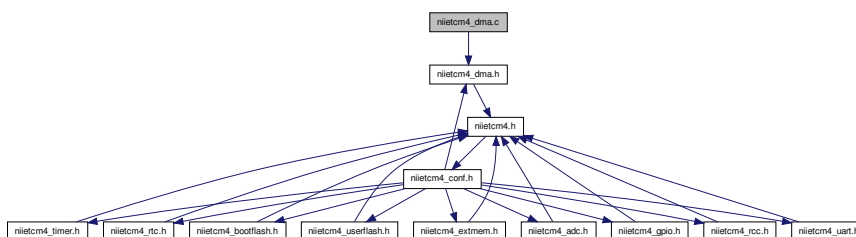
© 2015 ОАО "НИИЭТ"

8.7 Файл niietcm4_dma.c

Файл содержит реализацию всех функции для работы с DMA.

```
#include "niietcm4_dma.h"
```

Граф включаемых заголовочных файлов для niietcm4_dma.c:



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.

8.7.1 Подробное описание

Файл содержит реализацию всех функции для работы с DMA.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

10.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С 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"

8.7.2 Функции

8.7.2.1 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

8.7.2.2 void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)

Деинициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 87

Перекрестные ссылки DMA_Channel_TypeDef::CHANNEL_CFG, DMA_Channel_TypeDef::DST↔_DATA_END и DMA_Channel_TypeDef::SRC_DATA_END.

8.7.2.3 void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)

Активация каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

```
8.7.2.4 void DMA_ChannelInit ( DMA_Channel_TypeDef * DMA_Channel,
DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct )
```

Инициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHEABLE`, `DMA_Channel_TypeDef::CHANNEL_CFG_bit`, `_CHANNEL_CFG_bits::CYCLE_CTRL`, `DMA_ChannelInit_TypeDef::DMA_ArbitrationRate`, `DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr`, `DMA_ChannelInit_TypeDef::DMA_DstDataInc`, `DMA_ChannelInit_TypeDef::DMA_DstDataSize`, `DMA_ChannelInit_TypeDef::DMA_DstProtect`, `DMA_ChannelInit_TypeDef::DMA_Mode`, `DMA_ChannelInit_TypeDef::DMA_NextUseburst`, `DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr`, `DMA_ChannelInit_TypeDef::DMA_SrcDataInc`, `DMA_ChannelInit_TypeDef::DMA_SrcDataSize`, `DMA_ChannelInit_TypeDef::DMA_SrcProtect`, `DMA_ChannelInit_TypeDef::DMA_TransfersTotal`, `DMA_Channel_TypeDef::DST_DATA_END`, `_CHANNEL_CFG_bits::DST_INC`, `_CHANNEL_CFG_bits::DST_PROT_BUFFERABLE`, `_CHANNEL_CFG_bits::DST_PROT_CACHEABLE`, `_CHANNEL_CFG_bits::DST_PROT_PRIVILEGED`, `_CHANNEL_CFG_bits::DST_SIZE`, `IS_DMA_ARBITRATION_RATE`, `IS_DMA_DATA_INC`, `IS_DMA_DATA_SIZE`, `IS_DMA_MODE`, `IS_DMA_TRANSFERS_TOTAL`, `IS_FUNCTIONAL_STATE`, `_CHANNEL_CFG_bits::N_MINUS_1`, `_CHANNEL_CFG_bits::NEXT_USEBURST`, `DMA_Protect_TypeDef::PRIVILEGED`, `_CHANNEL_CFG_bits::R_POWER`, `DMA_Channel_TypeDef::SRC_DATA_END`, `_CHANNEL_CFG_bits::SRC_INC`, `_CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE`, `_CHANNEL_CFG_bits::SRC_PROT_CACHEABLE`, `_CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED` и `_CHANNEL_CFG_bits::SRC_SIZE`.

```
8.7.2.5 void DMA_ChannelStructInit ( DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct )
```

Заполнение каждого члена структуры `DMA_ChannelInitStruct` значениями по умолчанию.

Аргументы

DMA_↔ ChannelInit↔ Struct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которую необходимо проинициализировать.
---------------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 146

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CAC↔HEABLE, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ArbitrationRate_1, DM↔A_DataInc_Disable, DMA_DataSize_8, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, D↔MA_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_Type↔Def::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_Type↔Def::DMA_TransfersTotal и DMA_Protect_TypeDef::PRIVELGED.

8.7.2.6 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 524

8.7.2.7 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 174

8.7.2.8 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 503

8.7.2.9 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.7.2.10 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 DMA_Init_TypeDef , которая содержит конфигурационную информацию.
----------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 195

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

8.7.2.11 void DMA_MasterEnableCmd (FunctionalState State)

Разрешения работы контроллера DMA.

Внимание

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

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 287

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.7.2.12 FunctionalState DMA_MasterEnableStatus ()

Состояние контроллера DMA.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 455

8.7.2.13 void DMA_PrmAltCmd (uint32_t DMA_Channel, FunctionalState State)

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 397

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.7.2.14 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 DMA_Protect_TypeDef .
----------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 243

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, IS_FUNCTIONAL_STATE и DMA_Protect_TypeDef::PRIVELGED.

Используется в DMA_Init().

8.7.2.15 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 Маски каналов по номеру .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

8.7.2.16 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

8.7.2.17 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры `DMA_InitStruct` значениями по умолчанию.

Аргументы

DMA_InitStruct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
----------------	--

Возвращаемые значения

Нет

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

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHEABLE`, `DMA_Init_TypeDef::DMA_Channel`, `DMA_Init_TypeDef::DMA_ChannelEnable`, `DMA_Init_TypeDef::DMA_HighPriority`, `DMA_Init_TypeDef::DMA_PrmAlt`, `DMA_Init_TypeDef::DMA_Protection`, `DMA_Init_TypeDef::DMA_ReqMask`, `DMA_Init_TypeDef::DMA_UseBurst` и `DMA_Protect_TypeDef::PRIVELGED`.

8.7.2.18 void DMA_SWRequestCmd (uint32_t DMA_Channel)

Программный запрос на осуществление передач DMA по выбранным каналам.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 308

Перекрестные ссылки IS_DMA_CHANNEL.

8.7.2.19 void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)

Установка пакетного обмена каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 324

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.7.2.20 FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 478

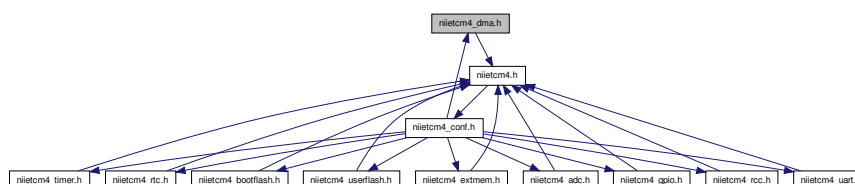
Перекрестные ссылки IS_GET_DMA_CHANNEL.

8.8 Файл niietcm4_dma.h

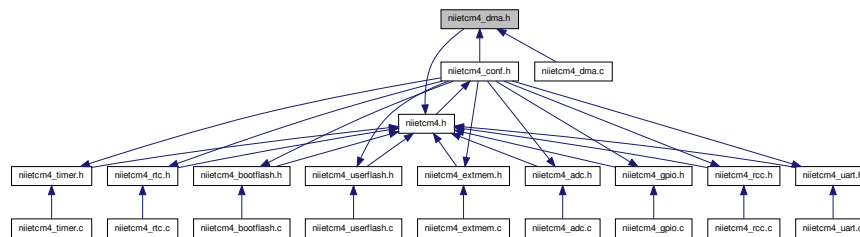
Файл содержит все прототипы функций для DMA.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_dma.h:



Граф файлов, в которые включается этот файл:



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)0x00003FFF)
- `#define` [CHANNEL_CFG_R_POWER_Msk](#) ((uint32_t)0x0003C000)
- `#define` [CHANNEL_CFG_SRC_PROT_CTRL_Msk](#) ((uint32_t)0x001C0000)
- `#define` [CHANNEL_CFG_DST_PROT_CTRL_Msk](#) ((uint32_t)0x00E00000)
- `#define` [CHANNEL_CFG_SRC_SIZE_Msk](#) ((uint32_t)0x03000000)
- `#define` [CHANNEL_CFG_SRC_INC_Msk](#) ((uint32_t)0x0C000000)
- `#define` [CHANNEL_CFG_DST_SIZE_Msk](#) ((uint32_t)0x30000000)
- `#define` [CHANNEL_CFG_DST_INC_Msk](#) ((uint32_t)0xC0000000)
- `#define` [DMA_Channel_All](#) ((uint32_t)0x00FFFFFF)

```

• #define DMA_Channel_UART0_TX ((uint32_t)0x00000001)
• #define DMA_Channel_UART1_TX ((uint32_t)0x00000002)
• #define DMA_Channel_UART2_TX ((uint32_t)0x00000004)
• #define DMA_Channel_UART3_TX ((uint32_t)0x00000008)
• #define DMA_Channel_UART0_RX ((uint32_t)0x00000010)
• #define DMA_Channel_UART1_RX ((uint32_t)0x00000020)
• #define DMA_Channel_UART2_RX ((uint32_t)0x00000040)
• #define DMA_Channel_UART3_RX ((uint32_t)0x00000080)
• #define DMA_Channel_ADCSEQ0 ((uint32_t)0x00000100)
• #define DMA_Channel_ADCSEQ1 ((uint32_t)0x00000200)
• #define DMA_Channel_ADCSEQ2 ((uint32_t)0x00000400)
• #define DMA_Channel_ADCSEQ3 ((uint32_t)0x00000800)
• #define DMA_Channel_ADCSEQ4 ((uint32_t)0x00001000)
• #define DMA_Channel_ADCSEQ5 ((uint32_t)0x00002000)
• #define DMA_Channel_ADCSEQ6 ((uint32_t)0x00004000)
• #define DMA_Channel_ADCSEQ7 ((uint32_t)0x00008000)
• #define DMA_Channel_SPI0_TX ((uint32_t)0x00010000)
• #define DMA_Channel_SPI1_TX ((uint32_t)0x00020000)
• #define DMA_Channel_SPI2_TX ((uint32_t)0x00040000)
• #define DMA_Channel_SPI3_TX ((uint32_t)0x00080000)
• #define DMA_Channel_SPI0_RX ((uint32_t)0x00100000)
• #define DMA_Channel_SPI1_RX ((uint32_t)0x00200000)
• #define DMA_Channel_SPI2_RX ((uint32_t)0x00400000)
• #define DMA_Channel_SPI3_RX ((uint32_t)0x00800000)
• #define DMA_Channel_0 ((uint32_t)0x00000001)
• #define DMA_Channel_1 ((uint32_t)0x00000002)
• #define DMA_Channel_2 ((uint32_t)0x00000004)
• #define DMA_Channel_3 ((uint32_t)0x00000008)
• #define DMA_Channel_4 ((uint32_t)0x00000010)
• #define DMA_Channel_5 ((uint32_t)0x00000020)
• #define DMA_Channel_6 ((uint32_t)0x00000040)
• #define DMA_Channel_7 ((uint32_t)0x00000080)
• #define DMA_Channel_8 ((uint32_t)0x00000100)
• #define DMA_Channel_9 ((uint32_t)0x00000200)
• #define DMA_Channel_10 ((uint32_t)0x00000400)
• #define DMA_Channel_11 ((uint32_t)0x00000800)
• #define DMA_Channel_12 ((uint32_t)0x00001000)
• #define DMA_Channel_13 ((uint32_t)0x00002000)
• #define DMA_Channel_14 ((uint32_t)0x00004000)
• #define DMA_Channel_15 ((uint32_t)0x00008000)
• #define DMA_Channel_16 ((uint32_t)0x00010000)
• #define DMA_Channel_17 ((uint32_t)0x00020000)
• #define DMA_Channel_18 ((uint32_t)0x00040000)
• #define DMA_Channel_19 ((uint32_t)0x00080000)
• #define DMA_Channel_20 ((uint32_t)0x00100000)
• #define DMA_Channel_21 ((uint32_t)0x00200000)
• #define DMA_Channel_22 ((uint32_t)0x00400000)
• #define DMA_Channel_23 ((uint32_t)0x00800000)
• #define IS_DMA_CHANNEL(CHANNEL) (((CHANNEL) != (uint32_t)0x000000) && (((CHANNEL) & (uint32_t)0xFF000000) == ((uint32_t)0x0000)))

    Макрос проверки маски каналов на попадание в допустимый диапазон.
• #define IS_GET_DMA_CHANNEL(CHANNEL)

    Макрос проверки маски канала при работе с каналами по отдельности.
• #define IS_DMA_MODE(MODE)

```

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

Перечисления

- `enum DMA_Mode_TypeDef {`
`DMA_Mode_Disable, DMA_Mode_Basic, DMA_Mode_AutoReq, DMA_Mode_PingPong,`
`DMA_Mode_PrmMemScatGath, DMA_Mode_AltMemScatGath, DMA_Mode_PrmPeriphScatGath, DMA_Mode_AltPeriphScatGath }`
 Выбор режима работы DMA.
- `enum DMA_ArbitrationRate_TypeDef {`
`DMA_ArbitrationRate_1, DMA_ArbitrationRate_2, DMA_ArbitrationRate_4, DMA_ArbitrationRate_8,`
`DMA_ArbitrationRate_16, DMA_ArbitrationRate_32, DMA_ArbitrationRate_64, DMA_ArbitrationRate_128,`
`DMA_ArbitrationRate_256, DMA_ArbitrationRate_512, DMA_ArbitrationRate_1024 }`
 Выбор количества передач до выполнения переарбитрации.
- `enum DMA_DataSize_TypeDef { DMA_DataSize_8, DMA_DataSize_16, DMA_DataSize_32 }`
 Разрядность данных источника или приемника
- `enum DMA_DataInc_TypeDef { DMA_DataInc_8, DMA_DataInc_16, DMA_DataInc_32, DMA_DataInc_Disable }`
 Шаг инкремента адреса источника при чтении или приемника при записи
- `enum DMA_State_TypeDef {`
`DMA_State_Free, DMA_State_ReadConfigData, DMA_State_ReadSrcDataEndPtr, DMA_State_ReadDstDataEndPtr,`
`DMA_State_ReadSrcData, DMA_State_WriteDstData, DMA_State_WaitReq, DMA_State_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 конечному автомату контроллера DMA.
- `FunctionalState` `DMA_MasterEnableStatus` ()
Состояние контроллера DMA.
- `FunctionalState` `DMA_WaitOnReqStatus` (`uint32_t` `DMA_Channel`)
Показывает поддерживает ли канал одиночные SREQ запросы.
- `OperationStatus` `DMA_ErrorStatus` ()
Показывает наличие ошибки на шине.
- void `DMA_ClearErrorStatus` ()
Сброс флага ошибки на шине.

8.8.1 Подробное описание

Файл содержит все прототипы функций для DMA.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

10.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.8.2 Макросы

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

Поле задания типа цикла DMA

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

8.8.2.2 `#define CHANNEL_CFG_CYCLE_CTRL_Pos 0`

Поле задания типа цикла DMA

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

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

Шаг инкремента адреса приемника

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

8.8.2.4 `#define CHANNEL_CFG_DST_INC_Pos 30`

Шаг инкремента адреса приемника

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

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

Защита шины АНВ-Lite при записи данных в приемник

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

8.8.2.6 `#define CHANNEL_CFG_DST_PROT_CTRL_Pos 21`

Защита шины АНВ-Lite при записи данных в приемник

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

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

Разрядность данных приемника

См. определение в файле niietcm4_dma.h строка 76

8.8.2.8 `#define CHANNEL_CFG_DST_SIZE_Pos 28`

Разрядность данных приемника

См. определение в файле niietcm4_dma.h строка 65

8.8.2.9 `#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

8.8.2.10 `#define CHANNEL_CFG_N_MINUS_1_Pos 4`

Общее количество передач в цикле работы

См. определение в файле niietcm4_dma.h строка 59

8.8.2.11 `#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

8.8.2.12 `#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

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

Количество передач до выполнения переарбитрации

См. определение в файле niietcm4_dma.h строка 71

8.8.2.14 `#define CHANNEL_CFG_R_POWER_Pos 14`

Количество передач до выполнения переарбитрации

См. определение в файле niietcm4_dma.h строка 60

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

Шаг инкремента адреса источника

См. определение в файле niietcm4_dma.h строка 75

8.8.2.16 `#define CHANNEL_CFG_SRC_INC_Pos 26`

Шаг инкремента адреса источника

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

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

Защита шины АHB-Lite при чтении данных из источника

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

8.8.2.18 `#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

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

Разрядность данных источника

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

8.8.2.20 `#define CHANNEL_CFG_SRC_SIZE_Pos 24`

Разрядность данных источника

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

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

Канал DMA 0

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

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

Канал DMA 1

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

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

Канал DMA 10

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

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

Канал DMA 11

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

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

Канал DMA 12

См. определение в файле niietcm4_dma.h строка 138

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

Канал DMA 13

См. определение в файле niietcm4_dma.h строка 139

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

Канал DMA 14

См. определение в файле niietcm4_dma.h строка 140

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

Канал DMA 15

См. определение в файле niietcm4_dma.h строка 141

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

Канал DMA 16

См. определение в файле niietcm4_dma.h строка 142

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

Канал DMA 17

См. определение в файле niietcm4_dma.h строка 143

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

Канал DMA 18

См. определение в файле niietcm4_dma.h строка 144

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

Канал DMA 19

См. определение в файле niietcm4_dma.h строка 145

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

Канал DMA 2

См. определение в файле niietcm4_dma.h строка 128

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

Канал DMA 20

См. определение в файле niietcm4_dma.h строка 146

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

Канал DMA 21

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

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

Канал DMA 22

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

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

Канал DMA 23

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

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

Канал DMA 3

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

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

Канал DMA 4

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

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

Канал DMA 5

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

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

Канал DMA 6

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

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

Канал DMA 7

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

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

Канал DMA 8

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

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

Канал DMA 9

См. определение в файле niietcm4_dma.h строка 135

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

Канал DMA секвенсора 0 АЦП

См. определение в файле niietcm4_dma.h строка 101

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

Канал DMA секвенсора 1 АЦП

См. определение в файле niietcm4_dma.h строка 102

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

Канал DMA секвенсора 2 АЦП

См. определение в файле niietcm4_dma.h строка 103

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

Канал DMA секвенсора 3 АЦП

См. определение в файле niietcm4_dma.h строка 104

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

Канал DMA секвенсора 4 АЦП

См. определение в файле niietcm4_dma.h строка 105

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

Канал DMA секвенсора 5 АЦП

См. определение в файле niietcm4_dma.h строка 106

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

Канал DMA секвенсора 6 АЦП

См. определение в файле niietcm4_dma.h строка 107

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

Канал DMA секвенсора 7 АЦП

См. определение в файле niietcm4_dma.h строка 108

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

Все каналы DMA

См. определение в файле niietcm4_dma.h строка 87

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

Канал DMA по приему от SPI0

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

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

Канал DMA по передаче от SPI0

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

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

Канал DMA по приему от SPI1

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

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

Канал DMA по передаче от SPI1

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

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

Канал DMA по приему от SPI2

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

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

Канал DMA по передаче от SPI2

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

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

Канал DMA по приему от SPI3

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

8.8.2.61 `#define DMA_Channel_SPI3_TX ((uint32_t)0x00080000)`

Канал DMA по передаче от SPI3

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

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

Канал DMA по приему от UART0

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

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

Канал DMA по передаче от UART0

См. определение в файле niietcm4_dma.h строка 93

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

Канал DMA по приему от UART1

См. определение в файле niietcm4_dma.h строка 98

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

Канал DMA по передаче от UART1

См. определение в файле niietcm4_dma.h строка 94

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

Канал DMA по приему от UART2

См. определение в файле niietcm4_dma.h строка 99

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

Канал DMA по передаче от UART2

См. определение в файле niietcm4_dma.h строка 95

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

Канал DMA по приему от UART3

См. определение в файле niietcm4_dma.h строка 100

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

Канал DMA по передаче от UART3

См. определение в файле niietcm4_dma.h строка 96

8.8.2.70 `#define IS_DMA_ARBITRATION_RATE(ARBITRATION_RATE)`

Макроопределение:

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

Макрос проверки аргументов типа [DMA_ArbitrationRate_TypeDef](#).

См. определение в файле `niietcm4_dma.h` строка 316

Используется в `DMA_ChannelInit()`.

8.8.2.71 `#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()`.

8.8.2.72 `#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()`.

8.8.2.73 `#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()`.

8.8.2.74 `#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 458

8.8.2.75 `#define IS_GET_DMA_CHANNEL(CHANNEL)`

Макроопределение:

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

Макрос проверки маски канала при работе с каналами по отдельности.

См. определение в файле niietcm4_dma.h строка 166

Используется в `DMA_WaitOnReqStatus()`.

8.8.3 Перечисления

8.8.3.1 `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

8.8.3.2 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

8.8.3.3 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

8.8.3.4 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

8.8.3.5 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_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

8.8.4 Функции

8.8.4.1 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

8.8.4.2 void DMA_ChannelDeInit (DMA_Channel_TypeDef * DMA_Channel)

Деинициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 87

Перекрестные ссылки DMA_Channel_TypeDef::CHANNEL_CFG, DMA_Channel_TypeDef::DST↔_DATA_END и DMA_Channel_TypeDef::SRC_DATA_END.

8.8.4.3 void DMA_ChannelEnableCmd (uint32_t DMA_Channel, FunctionalState State)

Активация каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_DMA_CHANNEL` и `IS_FUNCTIONAL_STATE`.

Используется в `DMA_Init()`.

```
8.8.4.4 void DMA_ChannelInit ( DMA_Channel_TypeDef * DMA_Channel,
DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct )
```

Инициализация канала DMA.

Аргументы

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

Возвращаемые значения

Нет

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

Перекрестные ссылки `DMA_Protect_TypeDef::BUFFERABLE`, `DMA_Protect_TypeDef::CACHEABLE`, `DMA_Channel_TypeDef::CHANNEL_CFG_bit`, `_CHANNEL_CFG_bits::CYCLE_CTRL`, `DMA_ChannelInit_TypeDef::DMA_ArbitrationRate`, `DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr`, `DMA_ChannelInit_TypeDef::DMA_DstDataInc`, `DMA_ChannelInit_TypeDef::DMA_DstDataSize`, `DMA_ChannelInit_TypeDef::DMA_DstProtect`, `DMA_ChannelInit_TypeDef::DMA_Mode`, `DMA_ChannelInit_TypeDef::DMA_NextUseburst`, `DMA_ChannelInit_TypeDef::DMA_SrcDataEndPtr`, `DMA_ChannelInit_TypeDef::DMA_SrcDataInc`, `DMA_ChannelInit_TypeDef::DMA_SrcDataSize`, `DMA_ChannelInit_TypeDef::DMA_SrcProtect`, `DMA_ChannelInit_TypeDef::DMA_TransfersTotal`, `DMA_Channel_TypeDef::DST_DATA_END`, `_CHANNEL_CFG_bits::DST_INC`, `_CHANNEL_CFG_bits::DST_PROT_BUFFERABLE`, `_CHANNEL_CFG_bits::DST_PROT_CACHEABLE`, `_CHANNEL_CFG_bits::DST_PROT_PRIVILEGED`, `_CHANNEL_CFG_bits::DST_SIZE`, `IS_DMA_ARBITRATION_RATE`, `IS_DMA_DATA_INC`, `IS_DMA_DATA_SIZE`, `IS_DMA_MODE`, `IS_DMA_TRANSFERS_TOTAL`, `IS_FUNCTIONAL_STATE`, `_CHANNEL_CFG_bits::N_MINUS_1`, `_CHANNEL_CFG_bits::NEXT_USEBURST`, `DMA_Protect_TypeDef::PRIVILEGED`, `_CHANNEL_CFG_bits::R_POWER`, `DMA_Channel_TypeDef::SRC_DATA_END`, `_CHANNEL_CFG_bits::SRC_INC`, `_CHANNEL_CFG_bits::SRC_PROT_BUFFERABLE`, `_CHANNEL_CFG_bits::SRC_PROT_CACHEABLE`, `_CHANNEL_CFG_bits::SRC_PROT_PRIVILEGED` и `_CHANNEL_CFG_bits::SRC_SIZE`.

```
8.8.4.5 void DMA_ChannelStructInit ( DMA_ChannelInit_TypeDef * DMA_ChannelInitStruct )
```

Заполнение каждого члена структуры `DMA_ChannelInitStruct` значениями по умолчанию.

Аргументы

DMA_↔ ChannelInit↔ Struct	Указатель на структуру типа DMA_ChannelInit_TypeDef , которую необходимо проинициализировать.
---------------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 146

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CAC↔HEABLE, DMA_ChannelInit_TypeDef::DMA_ArbitrationRate, DMA_ArbitrationRate_1, DM↔A_DataInc_Disable, DMA_DataSize_8, DMA_ChannelInit_TypeDef::DMA_DstDataEndPtr, D↔MA_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_Type↔Def::DMA_SrcDataSize, DMA_ChannelInit_TypeDef::DMA_SrcProtect, DMA_ChannelInit_Type↔Def::DMA_TransfersTotal и DMA_Protect_TypeDef::PRIVELGED.

8.8.4.6 void DMA_ClearErrorStatus ()

Сброс флага ошибки на шине.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 524

8.8.4.7 void DMA_DeInit ()

Деинициализация контроллера DMA.

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 174

8.8.4.8 OperationStatus DMA_ErrorStatus ()

Показывает наличие ошибки на шине.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 503

8.8.4.9 void DMA_HighPriorityCmd (uint32_t DMA_Channel, FunctionalState State)

Установка высокого приоритета каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 421

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.8.4.10 void DMA_Init (DMA_Init_TypeDef * DMA_InitStruct)

Инициализация контроллера DMA.

Внимание

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

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 195

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

8.8.4.11 void DMA_MasterEnableCmd (FunctionalState State)

Разрешения работы контроллера DMA.

Внимание

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

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 287

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.8.4.12 FunctionalState DMA_MasterEnableStatus ()

Состояние контроллера DMA.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 455

8.8.4.13 void DMA_PrmAltCmd (uint32_t DMA_Channel, FunctionalState State)

Установка первичной/альтернативной управляющей структуры каналов DMA.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 397

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.8.4.14 void DMA_ProtectionConfig (DMA_Protect_TypeDef * DMA_Protection)

Управление защитой шины при обращении DMA к управляющим данным.

Аргументы

DMA_Protection	Структура, содержащая конфигурацию защиты. Параметр принимает структуру типа DMA_Protect_TypeDef .
----------------	--

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_dma.c строка 243

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, IS_FUNCTIONAL_STATE и DMA_Protect_TypeDef::PRIVELGED.

Используется в DMA_Init().

8.8.4.15 void DMA_ReqMaskCmd (uint32_t DMA_Channel, FunctionalState State)

Маскирование каналов DMA.

Внимание

По маскированным каналам игнорируются запросы на передачи.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 349

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.8.4.16 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

8.8.4.17 void DMA_StructInit (DMA_Init_TypeDef * DMA_InitStruct)

Заполнение каждого члена структуры DMA_InitStruct значениями по умолчанию.

Аргументы

DMA_InitStruct	Указатель на структуру типа DMA_Init_TypeDef , которую необходимо проинициализировать.
----------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 212

Перекрестные ссылки DMA_Protect_TypeDef::BUFFERABLE, DMA_Protect_TypeDef::CACHEABLE, DMA_Init_TypeDef::DMA_Channel, DMA_Init_TypeDef::DMA_ChannelEnable, DMA_Init_TypeDef::DMA_HighPriority, DMA_Init_TypeDef::DMA_PrmAlt, DMA_Init_TypeDef::DMA_Protection, DMA_Init_TypeDef::DMA_ReqMask, DMA_Init_TypeDef::DMA_UseBurst и DMA_Protect_TypeDef::PRIVELGED.

8.8.4.18 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 Маски каналов по номеру .
-------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 308

Перекрестные ссылки IS_DMA_CHANNEL.

8.8.4.19 void DMA_UseBurstCmd (uint32_t DMA_Channel, FunctionalState State)

Установка пакетного обмена каналов DMA.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_dma.c строка 324

Перекрестные ссылки IS_DMA_CHANNEL и IS_FUNCTIONAL_STATE.

Используется в DMA_Init().

8.8.4.20 FunctionalState DMA_WaitOnReqStatus (uint32_t DMA_Channel)

Показывает поддерживает ли канал одиночные SREQ запросы.

Возвращаемые значения

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

См. определение в файле niietcm4_dma.c строка 478

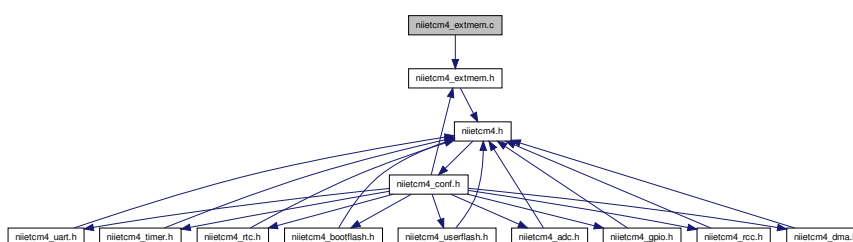
Перекрестные ссылки IS_GET_DMA_CHANNEL.

8.9 Файл niietcm4_extmem.c

Файл содержит реализацию всех функции для работы с интерфейсом внешней памяти.

```
#include "niietcm4_extmem.h"
```

Граф включаемых заголовочных файлов для niietcm4_extmem.c:



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.

8.9.1 Подробное описание

Файл содержит реализацию всех функций для работы с интерфейсом внешней памяти.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

08.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.9.2 Макросы

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

Значение по сбросу регистра EXT_MEM_CFG

См. определение в файле `nietcm4_extmem.c` строка 64

Используется в `EXTMEM_DeInit()`.

8.9.3 Функции

8.9.3.1 `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`.

8.9.3.2 `void EXTMEM_Init (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)`

Инициализирует внешнюю память согласно параметрам структуры `EXTMEM_InitStruct`.

Аргументы

<code>EXTMEM_InitStruct</code>	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
--------------------------------	---

Возвращаемые значения

Нет

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

Перекрестные ссылки `IS_EXTMEM_CE_MASK`, `IS_EXTMEM_READ_WAITSTATE`, `IS_EXTMEM_RW_WAITSTATE`, `IS_EXTMEM_WIDTH` и `IS_EXTMEM_WRITE_WAITSTATE`.

8.9.3.3 `void EXTMEM_StructInit (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)`

Заполнение каждого члена структуры `EXTMEM_InitStruct` значениями по умолчанию.

Аргументы

<code>EXTMEM_InitStruct</code>	Указатель на структуру типа EXTMEM_Init_TypeDef , которую необходимо проинициализировать.
--------------------------------	---

Возвращаемые значения

Нет

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

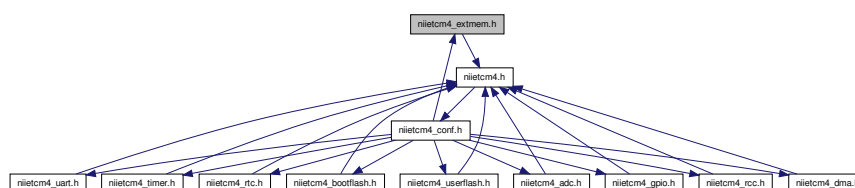
Перекрестные ссылки `EXTMEM_Init_TypeDef::CEMask`, `EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState`, `EXTMEM_ReadWaitState_8`, `EXTMEM_Init_TypeDef::EXTMEM_RWWaitState`, `EXTMEM_RWWaitState_1`, `EXTMEM_Init_TypeDef::EXTMEM_Width`, `EXTMEM_Width_16bit`, `EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState` и `EXTMEM_WriteWaitState_1`.

8.10 Файл `niietcm4_extmem.h`

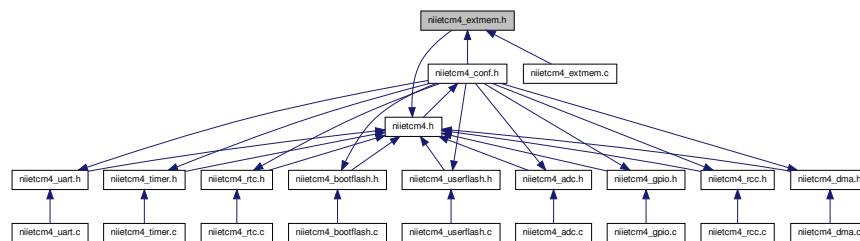
Файл содержит все прототипы функций для интерфейса внешней памяти.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для `niietcm4_extmem.h`:



Граф файлов, в которые включается этот файл:



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.

8.10.1 Подробное описание

Файл содержит все прототипы функций для интерфейса внешней памяти.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

08.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.10.2 Макросы

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

Маска бита 11 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 56

8.10.2.2 `#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

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

Маска бита 12 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 57

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

Маска бита 13 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 58

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

Маска бита 14 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 59

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

Маска бита 15 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 60

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

Маска бита 16 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 61

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

Маска бита 17 адреса контроллера внешней памяти.

См. определение в файле niietcm4_extmem.h строка 62

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

Маска бита 18 адреса контроллера внешней памяти.

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

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

Маска бита 19 адреса контроллера внешней памяти.

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

8.10.2.11 `#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()`.

8.10.2.12 `#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()`.

8.10.2.13 `#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()`.

8.10.2.14 #define IS_EXTMEM_WRITE_WAITSTATE(WAITSTATE)

Макроопределение:

```
((WAITSTATE) == EXTMEM_WriteWaitState_1) || \
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().

8.10.3 Перечисления

8.10.3.1 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

8.10.3.2 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

8.10.3.3 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

8.10.3.4 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

8.10.4 Функции

8.10.4.1 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.

8.10.4.2 void EXTMEM_Init (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Инициализирует внешнюю память согласно параметрам структуры EXTMEM_InitStruct.

Аргументы

EXTMEM_↔ InitStruct	Указатель на структуру типа EXTMEM_Init_TypeDef , которая содержит конфигурационную информацию.
------------------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_extmem.c строка 85

Перекрестные ссылки IS_EXTMEM_CE_MASK, IS_EXTMEM_READ_WAITSTATE, IS_EXTMEM_RW_WAITSTATE, IS_EXTMEM_WIDTH и IS_EXTMEM_WRITE_WAITSTATE.

8.10.4.3 void EXTMEM_StructInit (EXTMEM_Init_TypeDef * EXTMEM_InitStruct)

Заполнение каждого члена структуры EXTMEM_InitStruct значениями по умолчанию.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_extmem.c строка 107

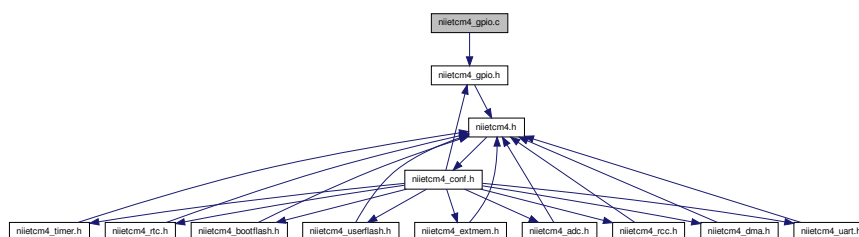
Перекрестные ссылки EXTMEM_Init_TypeDef::CEMask, EXTMEM_Init_TypeDef::EXTMEM_ReadWaitState, EXTMEM_ReadWaitState_8, EXTMEM_Init_TypeDef::EXTMEM_RWWaitState, EXTMEM_RWWaitState_1, EXTMEM_Init_TypeDef::EXTMEM_Width, EXTMEM_Width_16bit, EXTMEM_Init_TypeDef::EXTMEM_WriteWaitState и EXTMEM_WriteWaitState_1.

8.11 Файл niietcm4_gpio.c

Файл содержит реализацию всех функции для работы с модулями GPIO.

```
#include "niietcm4_gpio.h"
```

Граф включаемых заголовочных файлов для niietcm4_gpio.c:



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_GPIOCTLx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_GPIOQPx_Reset_Value ((uint32_t)0x00000000)`
- `#define GPIO_Regs_A_C_E_G_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_B_D_F_H_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOA_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOB_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOC_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOD_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOE_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOF_Mask ((uint32_t)0xFFFF0000)`
- `#define GPIO_Regs_GPIOG_Mask ((uint32_t)0x0000FFFF)`
- `#define GPIO_Regs_GPIOH_Mask ((uint32_t)0xFFFF0000)`

Функции

- `void GPIO_DeInit (NT_GPIO_TypeDef *GPIOx)`
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- `void GPIO_AltFuncConfig (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_AltFunc_TypeDef GPIO_AltFunc)`
- `void GPIO_Init (NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)`
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- `void GPIO_StructInit (GPIO_Init_TypeDef *GPIO_InitStruct)`
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.
- `uint32_t GPIO_ReadBit (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Чтение состояния выбранного пина.
- `uint32_t GPIO_Read (NT_GPIO_TypeDef *GPIOx)`
Чтение состояния выбранного порта GPIOx.
- `uint32_t GPIO_ReadMask (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)`
Чтение состояния выбранного порта GPIOx с использованием маски.
- `void GPIO_WriteBit (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, BitAction BitVal)`
Изменение состояния выбранного пина.
- `void GPIO_Write (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)`
Изменение состояния выбранного порта GPIOx.
- `void GPIO_WriteMask (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)`
Изменение состояния выбранного порта GPIOx с использованием маски.
- `void GPIO_SetBits (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Установка выбранных пинов.
- `void GPIO_ClearBits (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Сброс выбранных пинов.
- `void GPIO_ToggleBits (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)`
Переключение выбранных пинов в противоположное состояние.
- `void GPIO_QualConfig (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)`
Настройка фильтра выбранных пинов.
- `void GPIO_QualCmd (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)`
Включение входных фильтров.
- `void GPIO_SyncCmd (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, FunctionalState State)`
Включение пересинхронизации входов через 2 триггера-защелки.
- `void GPIO_ITConfig (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol)`

- Настройка прерываний пинов.
 - void `GPIO_ITCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, `FunctionalState` State)
- Включение прерываний выбранных пинов.
 - void `GPIO_ITStatusClear` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
- Очистка флагов прерываний выбранных пинов.

8.11.1 Подробное описание

Файл содержит реализацию всех функции для работы с модулями GPIO.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

26.10.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.11.2 Макросы

8.11.2.1 `#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()`.

8.11.2.2 `#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()`.

8.11.2.3 `#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()`.

8.11.2.4 `#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()`.

8.11.2.5 `#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()`.

8.11.2.6 `#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()`.

8.11.2.7 `#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()`.

8.11.2.8 `#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()`.

8.11.2.9 `#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()`.

8.11.2.10 `#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()`.

8.11.2.11 `#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().

8.11.2.12 `#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().

8.11.2.13 `#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().

8.11.2.14 `#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().

8.11.2.15 `#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().

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

Маска регистров для порта GPIOA

См. определение в файле niietcm4_gpio.c строка 96

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

Маска регистров для порта GPIOB

См. определение в файле niietcm4_gpio.c строка 97

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

Маска регистров для порта GPIOC

См. определение в файле niietcm4_gpio.c строка 98

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

Маска регистров для порта GPIOD

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

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

Маска регистров для порта GPIOE

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

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

Маска регистров для порта GPIOF

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

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

Маска регистров для порта GPIOG

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

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

Маска регистров для порта GPIOH

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

8.11.3 Функции

8.11.3.1 `void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)`

Сброс выбранных пинов.

Аргументы

<code>GPIOx</code>	Выбор порта, где x лежит в диапазоне А..Н.
<code>GPIO_Pin</code>	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 626

Перекрестные ссылки `IS_GPIO_ALL_PERIPH` и `IS_GPIO_PIN`.

8.11.3.2 `void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)`

Устанавливает все регистры выбранного `GPIOx` значениями по умолчанию.

Аргументы

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

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.

8.11.3.3 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 331

Перекрестные ссылки GPIO_Init_TypeDef::GPIO_AltFunc, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_In, GPIO_Dir_Out, GPIO_Init_TypeDef::GPIO_Load, GPIO_Load_16mA, GPIO_Load_8mA, GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Mode_IO, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_Dis, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_OutMode, GPIO_OutMode_OD, GPIO_OutMode_PP, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Init_TypeDef::GPIO_PullUp, GPIO_PullUp_Dis, GPIO_PullUp_En, IS_GPIO_ALL_PERIPH, IS_GPIO_ALT_FUNC, IS_GPIO_DIR, IS_GPIO_LOAD, IS_GPIO_MODE, IS_GPIO_OUT, IS_GPIO_OUT_MODE, IS_GPIO_PIN и IS_GPIO_PULLUP.

Используется в RCC_SysClkDiv2Out().

8.11.3.4 void GPIO_ITCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 913

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

```
8.11.3.5 void GPIO_ITConfig ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,  
GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol )
```

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 876

Перекрестные ссылки GPIO_IntPol_Neg, GPIO_IntPol_Pos, GPIO_IntType_Edge, GPIO_IntType_Level, IS_GPIO_ALL_PERIPH, IS_GPIO_INT_POL, IS_GPIO_INT_TYPE и IS_GPIO_PIN.

8.11.3.6 void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 938

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.11.3.7 void GPIO_QualCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.11.3.8 void GPIO_QualConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePerod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 664

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

8.11.3.9 uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)

Чтение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта GPIOx

См. определение в файле niietcm4_gpio.c строка 503

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.11.3.10 uint32_t GPIO_ReadBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина

См. определение в файле niietcm4_gpio.c строка 477

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN и IS_GPIO_ALL_PERIPH.

8.11.3.11 uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)

Чтение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски чтения.

Возвращаемые значения

Состояние порта GPIOx с учетом маски	
---	--

См. определение в файле niietcm4_gpio.c строка 518

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.11.3.12 void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 609

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.11.3.13 void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)

Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 456

Перекрестные ссылки GPIO_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().

8.11.3.14 void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 814

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.11.3.15 void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Переключение выбранных пинов в противоположное состояние.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 643

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.11.3.16 void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.11.3.17 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 546

Перекрестные ссылки Bit_CLEAR, Bit_SET, IS_GET_GPIO_PIN, IS_GPIO_ALL_PERIPH и I<S>S_GPIO_BIT_ACTION.

8.11.3.18 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне А..Н.
MaskVal	Значение маски.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 586

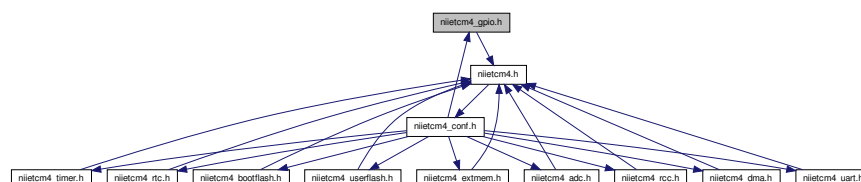
Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.12 Файл niietcm4_gpio.h

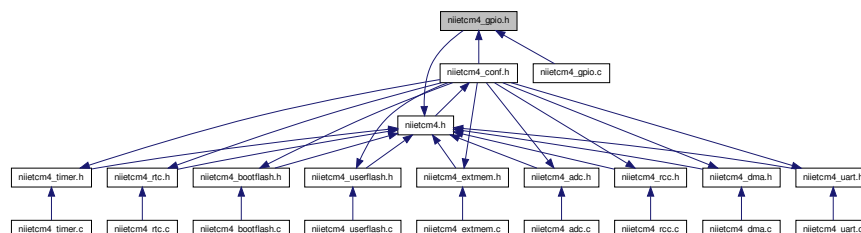
Файл содержит все прототипы функций для GPIO.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_gpio.h:



Граф файлов, в которые включается этот файл:



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) <= 0x0000) == ((uint32_t)0x0000))`
Макрос проверки номеров пинов на попадание в допустимый диапазон.
- `#define IS_GET_GPIO_PIN(PIN)`
Макрос проверки номера пина при работе с пинами по отдельности.

Перечисления

- `enum BitAction { Bit_CLEAR = 0, Bit_SET }`
Тип, определяющий состояния бита.
- `enum GPIO_Dir_TypeDef { GPIO_Dir_In, GPIO_Dir_Out }`
Выбор направления работы пина.
- `enum GPIO_Mode_TypeDef { GPIO_Mode_IO, GPIO_Mode_AltFunc }`
Выбор режима работы пина.
- `enum GPIO_IntType_TypeDef { GPIO_IntType_Level, GPIO_IntType_Edge }`
Выбор события для возникновения прерывания.
- `enum GPIO_IntPol_TypeDef { GPIO_IntPol_Neg, GPIO_IntPol_Pos }`
Выбор полярности события для возникновения прерывания.
- `enum GPIO_Out_TypeDef { GPIO_Out_Dis, GPIO_Out_En }`
Включение выхода пина.
- `enum GPIO_Load_TypeDef { GPIO_Load_8mA, GPIO_Load_16mA }`
Выбор максимальной нагрузочной способности пина.
- `enum GPIO_OutMode_TypeDef { GPIO_OutMode_PP, GPIO_OutMode_OD }`
Выбор режима работы выходных каскадов.
- `enum GPIO_PullUp_TypeDef { GPIO_PullUp_Dis, GPIO_PullUp_En }`
Включение подтяжки к питанию.
- `enum GPIO_Sync_TypeDef { GPIO_Sync_Dis, GPIO_Sync_En }`
Включение режима пересинхронизации входов через 2 триггера-защелки.
- `enum GPIO_Qual_TypeDef { GPIO_Qual_Dis, GPIO_Qual_En }`
Включение входного фильтра.
- `enum GPIO_QualMode_TypeDef { GPIO_QualMode_3sample, GPIO_QualMode_6sample }`
Выбор режима работы входного фильтра.
- `enum GPIO_AltFunc_TypeDef { GPIO_AltFunc_1, GPIO_AltFunc_2, GPIO_AltFunc_3 }`
Выбор номера альтернативной функции пина.

Функции

- `void GPIO_DeInit (NT_GPIO_TypeDef *GPIOx)`
Устанавливает все регистры выбранного GPIOx значениями по умолчанию.
- `void GPIO_Init (NT_GPIO_TypeDef *GPIOx, GPIO_Init_TypeDef *GPIO_InitStruct)`
Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.
- `void GPIO_StructInit (GPIO_Init_TypeDef *GPIO_InitStruct)`
Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

- void `GPIO_AltFuncConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_AltFunc_TypeDef](#) GPIO_AltFunc)
- uint32_t `GPIO_ReadBit` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Чтение состояния выбранного пина.
- uint32_t `GPIO_Read` (NT_GPIO_TypeDef *GPIOx)
Чтение состояния выбранного порта GPIOx.
- uint32_t `GPIO_ReadMask` (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal)
Чтение состояния выбранного порта GPIOx с использованием маски.
- void `GPIO_WriteBit` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [BitAction](#) BitVal)
Изменение состояния выбранного пина.
- void `GPIO_Write` (NT_GPIO_TypeDef *GPIOx, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx.
- void `GPIO_WriteMask` (NT_GPIO_TypeDef *GPIOx, uint32_t MaskVal, uint32_t PortVal)
Изменение состояния выбранного порта GPIOx с использованием маски.
- void `GPIO_SetBits` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Установка выбранных пинов.
- void `GPIO_ClearBits` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Сброс выбранных пинов.
- void `GPIO_ToggleBits` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Переключение выбранных пинов в противоположное состояние.
- void `GPIO_QualConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_QualMode_TypeDef](#) Mode, uint32_t SamplePerod)
Настройка фильтра выбранных пинов.
- void `GPIO_QualCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение входных фильтров.
- void `GPIO_SyncCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение пересинхронизации входов через 2 триггера-защелки.
- void `GPIO_ITConfig` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [GPIO_IntType_TypeDef](#) IntType, [GPIO_IntPol_TypeDef](#) IntPol)
Настройка прерываний пинов.
- void `GPIO_ITCmd` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin, [FunctionalState](#) State)
Включение прерываний выбранных пинов.
- void `GPIO_ITStatusClear` (NT_GPIO_TypeDef *GPIOx, uint32_t GPIO_Pin)
Очистка флагов прерываний выбранных пинов.

8.12.1 Подробное описание

Файл содержит все прототипы функций для GPIO.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), 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"

8.12.2 Макросы

8.12.2.1 `#define GPIO_Pin_0 ((uint32_t)0x0001)`

Пин 0 выбран.

См. определение в файле niietcm4_gpio.h строка 319

Используется в `RCC_SysClkDiv2Out()`.

8.12.2.2 `#define GPIO_Pin_0_3 ((uint32_t)0x000F)`

Пины 0-3 выбраны.

См. определение в файле niietcm4_gpio.h строка 335

8.12.2.3 `#define GPIO_Pin_0_7 ((uint32_t)0x00FF)`

Пины 0-7 выбраны.

См. определение в файле niietcm4_gpio.h строка 339

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

Пин 1 выбран.

См. определение в файле niietcm4_gpio.h строка 320

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

Пин 10 выбран.

См. определение в файле niietcm4_gpio.h строка 329

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

Пин 11 выбран.

См. определение в файле niietcm4_gpio.h строка 330

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

Пин 12 выбран.

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

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

Пины 12-15 выбраны.

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

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

Пин 13 выбран.

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

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

Пин 14 выбран.

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

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

Пин 15 выбран.

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

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

Пин 2 выбран.

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

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

Пин 3 выбран.

См. определение в файле `niietcm4_gpio.h` строка 322

8.12.2.14 `#define GPIO_Pin_4 ((uint32_t)0x0010)`

Пин 4 выбран.

См. определение в файле `niietcm4_gpio.h` строка 323

8.12.2.15 `#define GPIO_Pin_4_7 ((uint32_t)0x00F0)`

Пины 4-7 выбраны.

См. определение в файле `niietcm4_gpio.h` строка 336

8.12.2.16 `#define GPIO_Pin_5 ((uint32_t)0x0020)`

Пин 5 выбран.

См. определение в файле niietcm4_gpio.h строка 324

8.12.2.17 `#define GPIO_Pin_6 ((uint32_t)0x0040)`

Пин 6 выбран.

См. определение в файле niietcm4_gpio.h строка 325

8.12.2.18 `#define GPIO_Pin_7 ((uint32_t)0x0080)`

Пин 7 выбран.

См. определение в файле niietcm4_gpio.h строка 326

8.12.2.19 `#define GPIO_Pin_8 ((uint32_t)0x0100)`

Пин 8 выбран.

См. определение в файле niietcm4_gpio.h строка 327

8.12.2.20 `#define GPIO_Pin_8_11 ((uint32_t)0x0F00)`

Пины 8-11 выбраны.

См. определение в файле niietcm4_gpio.h строка 337

8.12.2.21 `#define GPIO_Pin_8_15 ((uint32_t)0xFF00)`

Пины 8-15 выбраны.

См. определение в файле niietcm4_gpio.h строка 340

8.12.2.22 `#define GPIO_Pin_9 ((uint32_t)0x0200)`

Пин 9 выбран.

См. определение в файле niietcm4_gpio.h строка 328

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

Все пины выбраны.

См. определение в файле niietcm4_gpio.h строка 341

Используется в `GPIO_StructInit()`.

8.12.2.24 `#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()`.

8.12.2.25 `#define IS_GPIO_ALT_FUNC(ALT_FUNC)`

Макроопределение:

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

Макрос проверки аргументов типа `GPIO_AltFunc_TypeDef`.

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

Используется в `GPIO_Init()`.

8.12.2.26 `#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()`.

8.12.2.27 `#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()`.

8.12.2.28 `#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()`.

8.12.2.29 `#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()`.

8.12.2.30 `#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()`.

8.12.2.31 `#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()`.

8.12.2.32 `#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()`.

8.12.2.33 `#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()`.

8.12.2.34 `#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

8.12.2.35 `#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()`.

8.12.2.36 `#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

8.12.3 Перечисления

8.12.3.1 `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

8.12.3.2 `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

8.12.3.3 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

8.12.3.4 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

8.12.3.5 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

8.12.3.6 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

8.12.3.7 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

8.12.3.8 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

8.12.3.9 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

8.12.3.10 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

8.12.3.11 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

8.12.3.12 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

8.12.3.13 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

8.12.4 Функции

8.12.4.1 void GPIO_ClearBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Сброс выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 626

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.12.4.2 void GPIO_DeInit (NT_GPIO_TypeDef * GPIOx)

Устанавливает все регистры выбранного GPIOx значениями по умолчанию.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 123

Перекрестные ссылки GPIO_DATAOUT_Reset_Value, GPIO_GPIODEN0_Reset_Value, GPIO_GPIODEN1_Reset_Value, GPIO_GPIODEN2_Reset_Value, GPIO_GPIODEN3_Reset_Value, GPIO_GPIOODCTLx_Reset_Value, GPIO_GPIOODSCTLx_Reset_Value, GPIO_GPIOPCTLx_Reset_Value, GPIO_GPIOPUCTLx_Reset_Value, GPIO_GPIOQEx_Reset_Value, GPIO_GPIOQMx_Reset_Value, GPIO_GPIOQPx_Reset_Value, GPIO_GPIOSEx_Reset_Value, GPIO_Regs_A_C_E_G_Mask, GPIO_Regs_B_D_F_H_Mask и IS_GPIO_ALL_PERIPH.

8.12.4.3 void GPIO_Init (NT_GPIO_TypeDef * GPIOx, GPIO_Init_TypeDef * GPIO_InitStruct)

Инициализирует модуль GPIOx согласно параметрам структуры GPIO_InitStruct.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Init↔ Struct	Указатель на структуру типа GPIO_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Нет

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

Перекрестные ссылки [GPIO_Init_TypeDef::GPIO_AltFunc](#), [GPIO_Init_TypeDef::GPIO_Dir](#), [GPIO_Dir_In](#), [GPIO_Dir_Out](#), [GPIO_Init_TypeDef::GPIO_Load](#), [GPIO_Load_16mA](#), [GPIO_Load_8mA](#), [GPIO_Init_TypeDef::GPIO_Mode](#), [GPIO_Mode_AltFunc](#), [GPIO_Mode_IO](#), [GPIO_Init_TypeDef::GPIO_Out](#), [GPIO_Out_Dis](#), [GPIO_Out_En](#), [GPIO_Init_TypeDef::GPIO_OutMode](#), [GPIO_OutMode_OD](#), [GPIO_OutMode_PP](#), [GPIO_Init_TypeDef::GPIO_Pin](#), [GPIO_Init_TypeDef::GPIO_PullUp](#), [GPIO_PullUp_Dis](#), [GPIO_PullUp_En](#), [IS_GPIO_ALL_PERIPH](#), [IS_GPIO_ALT_FUNC](#), [IS_GPIO_DIR](#), [IS_GPIO_LOAD](#), [IS_GPIO_MODE](#), [IS_GPIO_OUT](#), [IS_GPIO_OUT_MODE](#), [IS_GPIO_PIN](#) и [IS_GPIO_PULLUP](#).

Используется в `RCC_SysClkDiv2Out()`.

```
8.12.4.4 void GPIO_ITCmd ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,
FunctionalState State )
```

Включение прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 913

Перекрестные ссылки [IS_FUNCTIONAL_STATE](#), [IS_GPIO_ALL_PERIPH](#) и [IS_GPIO_PIN](#).

```
8.12.4.5 void GPIO_ITConfig ( NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin,
GPIO_IntType_TypeDef IntType, GPIO_IntPol_TypeDef IntPol )
```

Настройка прерываний пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.
IntType	Выбор события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntType_TypeDef .
IntPol	Выбор полярности события для возникновения прерывания. Параметр принимает любое значение из GPIO_IntPol_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 876

Перекрестные ссылки GPIO_IntPol_Neg, GPIO_IntPol_Pos, GPIO_IntType_Edge, GPIO_IntType_Level, IS_GPIO_ALL_PERIPH, IS_GPIO_INT_POL, IS_GPIO_INT_TYPE и IS_GPIO_PIN.

8.12.4.6 void GPIO_ITStatusClear (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Очистка флагов прерываний выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 938

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.12.4.7 void GPIO_QualCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение входных фильтров.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 753

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.12.4.8 void GPIO_QualConfig (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, GPIO_QualMode_TypeDef Mode, uint32_t SamplePeriod)

Настройка фильтра выбранных пинов.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Mode	Выбор режима работы. Параметр может принимать любое значение из GPIO_QualMode_TypeDef .
SamplePeriod	Количество тактов системной частоты между отсчетами фильтра. Параметр принимает любое значение из диапазоне 0...255.

Возвращаемые значения

Нет

См. определение в файле `niietcm4_gpio.c` строка 664

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

8.12.4.9 `uint32_t GPIO_Read (NT_GPIO_TypeDef * GPIOx)`

Чтение состояния выбранного порта `GPIOx`.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
-------	--

Возвращаемые значения

Состояние порта <code>GPIOx</code>

См. определение в файле `niietcm4_gpio.c` строка 503

Перекрестные ссылки `IS_GPIO_ALL_PERIPH`.

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

Чтение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из <code>GPIO_Pin_x</code> , где x лежит в диапазоне 0..15.

Возвращаемые значения

Состояние выбранного пина

См. определение в файле `niietcm4_gpio.c` строка 477

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

8.12.4.11 `uint32_t GPIO_ReadMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal)`

Чтение состояния выбранного порта `GPIOx` с использованием маски.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
MaskVal	Значение маски чтения.

Возвращаемые значения

Состояние порта GPIOx с учетом маски	
---	--

См. определение в файле niietcm4_gpio.c строка 518

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.12.4.12 void GPIO_SetBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Установка выбранных пинов.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 609

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.12.4.13 void GPIO_StructInit (GPIO_Init_TypeDef * GPIO_InitStruct)

Заполнение каждого члена структуры GPIO_InitStruct значениями по умолчанию.

Аргументы

GPIO_InitStruct	Указатель на структуру типа GPIO_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 456

Перекрестные ссылки GPIO_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().

8.12.4.14 void GPIO_SyncCmd (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, FunctionalState State)

Включение пересинхронизации входов через 2 триггера-защелки.

Аргументы

GPIOx	выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 814

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.12.4.15 void GPIO_ToggleBits (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin)

Переключение выбранных пинов в противоположное состояние.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 643

Перекрестные ссылки IS_GPIO_ALL_PERIPH и IS_GPIO_PIN.

8.12.4.16 void GPIO_Write (NT_GPIO_TypeDef * GPIOx, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
PortVal	Значение которое будет записано.

Возвращаемые значения

Нет

См. определение в файле niietcm4_gpio.c строка 570

Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.12.4.17 void GPIO_WriteBit (NT_GPIO_TypeDef * GPIOx, uint32_t GPIO_Pin, BitAction BitVal)

Изменение состояния выбранного пина.

Аргументы

GPIOx	Выбор порта, где x лежит в диапазоне A..H.
GPIO_Pin	Выбор пинов. Этот параметр может принимать любое значение из GPIO_Pin_x, где x лежит в диапазоне 0..15.
BitVal	Значение которое будет записано. Параметр может принимать любое значение из BitAction .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 546

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

8.12.4.18 void GPIO_WriteMask (NT_GPIO_TypeDef * GPIOx, uint32_t MaskVal, uint32_t PortVal)

Изменение состояния выбранного порта GPIOx с использованием маски.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_gpio.c строка 586

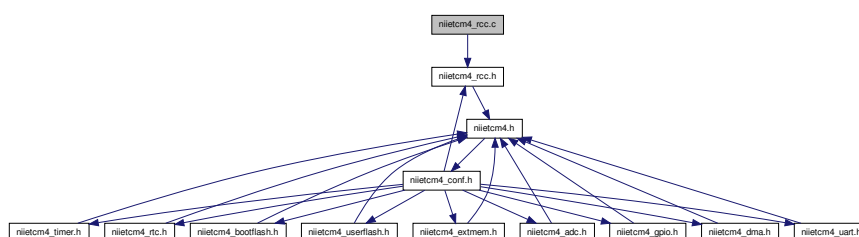
Перекрестные ссылки IS_GPIO_ALL_PERIPH.

8.13 Файл niietcm4_rcc.c

Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера.

```
#include "niietcm4_rcc.h"
```

Граф включаемых заголовочных файлов для niietcm4_rcc.c:



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 SysClkFreq)
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.

8.13.1 Подробное описание

Файл содержит реализацию всех функции для работы с тактированием и сбросом периферийных блоков микроконтроллера.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

06.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.13.2 Макросы

8.13.2.1 `#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().

8.13.2.2 `#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().

8.13.2.3 `#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().

8.13.2.4 `#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()`.

8.13.3 Функции

8.13.3.1 `void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)`

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 773

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`.

8.13.3.2 `void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 689

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`.

8.13.3.3 `void RCC_PeriphClkCmd (RCC_PeriphClk_TypeDef RCC_PeriphClk, FunctionalState State)`

Включение тактирования выбранного блока периферии.

Внимание

Блоки UART , SPI, ADC, USB управляются отдельно.

- [Тактирование UART](#)
- [Тактирование SPI](#)
- [Тактирование ADC](#)
- [Тактирование USB](#)

Аргументы

RCC_Periph← Clk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_← TypeDef.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 336

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_RCC_PERIPH_CLK.

8.13.3.4 void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_Periph← Rst	Выбор периферийного модуля. Параметр принимает любое значение из RCC← PeriphRst_TypeDef.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 863

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_PERIPH_RST, RCC_PeriphRst_E←
TH, RCC_PeriphRst_I2C0, RCC_PeriphRst_I2C1, RCC_PeriphRst_SPI0, RCC_PeriphRst_SPI1,
RCC_PeriphRst_SPI2, RCC_PeriphRst_SPI3, RCC_PeriphRst_Timer0, RCC_PeriphRst_Timer1,
RCC_PeriphRst_Timer2, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_←
UART2, RCC_PeriphRst_UART3, RCC_PeriphRst_USB и RCC_PeriphRst_WD.

Используется в UART_DeInit().

8.13.3.5 OperationStatus RCC_PLLAutoConfig (RCC_PLLRef_TypeDef RCC_PLLRef, uint32_t SysFreq)

Автоматическая конфигурация PLL для получения желаемой системной частоты.

С учетом данных об источнике частоты для PLL, а также о значении желаемой частоты, вычисляются все необходимые коэффициенты.

Внимание

Если Freq < 50 МГц, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

RCC_PLLRef	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из RCC_PLLRef_TypeDef .
SysFreq	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 142

Перекрестные ссылки EXT_OSC_VALUE, IS_RCC_PLL_REF, IS_RCC_SYS_FREQ, RCC_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().

8.13.3.6 void RCC_PLLDeInit ()

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 292

Перекрестные ссылки RCC_PLL_CTRL_Reset_Value, RCC_PLL_NF_Reset_Value, RCC_PLL_NR_Reset_Value и RCC_PLL_OD_Reset_Value.

8.13.3.7 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 FIN – входная частота PLL.

Внимание

При расчете коэффициентов деления PLL должны выполняться следующие условия:

- 3,2 МГц < FIN < 150 МГц,
- 800 КГц < FREF < 8МГц,
- 200 МГц < FVCO < 500МГц,

где частота фазового детектора FREF вычисляется по формуле:

$$\text{FREF} = \text{FIN} / (2 \times \text{NR}),$$

а частота FVCO вычисляется по формуле:

$$\text{FVCO} = \text{FIN} \times (\text{NF} / \text{NR})$$

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 256

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().

8.13.3.8 void RCC_PLLPowerDownCmd (FunctionalState State)

Управление режимом PowerDown PLL.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 307

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.13.3.9 void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 278

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.

8.13.3.10 void RCC_SPIClkCmd (NT_SPI_TypeDef * SPIx, FunctionalState State)

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 642

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_SPI_ALL_PERIPH.

```
8.13.3.11 void RCC_SPIClkDivConfig ( NT_SPI_TypeDef * SPIx, uint32_t DivVal,
FunctionalState DivState )
```

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 604

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_SPI_ALL_PERIPH.

```
8.13.3.12 void RCC_SPIClkSel ( NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk
)
```

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 563

Перекрестные ссылки IS_RCC_SPI_CLK и IS_SPI_ALL_PERIPH.

```
8.13.3.13 void RCC_SysClkDiv2Out ( FunctionalState State )
```

Включение генерации тактового сигнала с частотой равной половине системной на выводе H[0]. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState: <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 106

Перекрестные ссылки GPIO_InitTypeDef::GPIO_AltFunc, GPIO_AltFunc_2, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_Out, GPIO_Init(), GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_0, GPIO_StructInit() и IS_FUNCTIONAL_STATE.

8.13.3.14 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_TypeDef .
------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 359

Перекрестные ссылки IS_RCC_SYS_CLK и RCC_WaitClkChange().

Используется в RCC_PLLAutoConfig().

8.13.3.15 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле niietcm4_rcc.c строка 388

8.13.3.16 void RCC_UARTClkCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.13.3.17 void RCC_UARTClkDivConfig (NT_UART_TypeDef * UARTx, uint32_t DivVal, FunctionalState DivState)

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 482

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_CLK_DIV и IS_UART_ALL_PERIPH.

8.13.3.18 void RCC_UARTClkSel (NT_UART_TypeDef * UARTx, RCC_UARTClk_TypeDef RCC_UARTClk)

Настройка источника тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
RCC_UARTClk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 442

Перекрестные ссылки IS_RCC_UART_CLK и IS_UART_ALL_PERIPH.

8.13.3.19 void RCC_USBClkCmd (FunctionalState State)

Включение тактирования USB.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 419

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.13.3.20 void RCC_USBCLKConfig (RCC_USBCLK_TypeDef RCC_USBCLK,
RCC_USBFreq_TypeDef RCC_USBFreq)

Настройка источника тактового сигнала для USB.

Аргументы

RCC_USBCLK	Выбор источника тактирования. Параметр принимает любое значение из RCC_USBCLK_TypeDef .
RCC_USBFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 402

Перекрестные ссылки IS_RCC_USB_CLK и IS_RCC_USB_FREQ.

8.13.3.21 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.

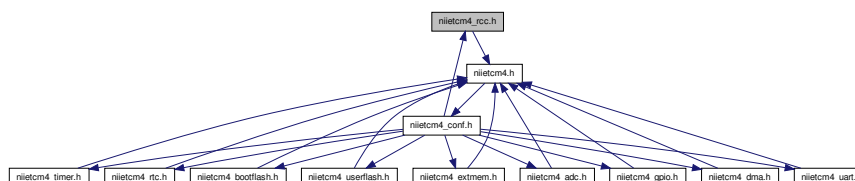
Используется в RCC_SysClkSel().

8.14 Файл niietcm4_rcc.h

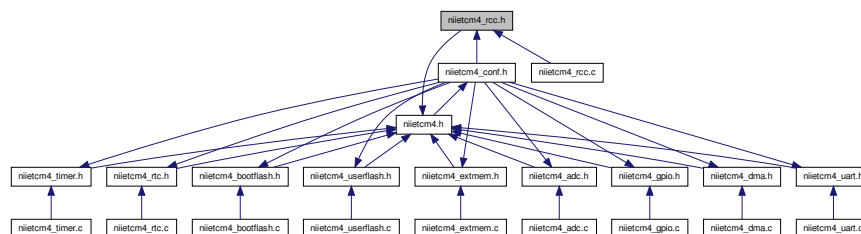
Файл содержит все прототипы функций для RCC (Reset & Clock Control).

#include "niietcm4.h"

Граф включаемых заголовочных файлов для niietcm4_rcc.h:



Граф файлов, в которые включается этот файл:



Структуры данных

- struct [RCC_PLLInit_TypeDef](#)
Структура инициализации PLL.

Макросы

- `#define RCC_CLK_CHANGE_TIMEOUT ((uint32_t)10000)`
- `#define IS_RCC_PLL_REF(PLL_REF)`
Макрос проверки аргументов типа [RCC_PLLRef_TypeDef](#).
- `#define IS_RCC_PLL_NO(PLL_NO)`
Макрос проверки аргументов типа [RCC_PLLNO_TypeDef](#).
- `#define IS_RCC_UART_CLK(UART_CLK)`
Макрос проверки аргументов типа [RCC_UARTClk_TypeDef](#).
- `#define IS_RCC_SPI_CLK(SPI_CLK)`
Макрос проверки аргументов типа [RCC_SPIClk_TypeDef](#).
- `#define IS_RCC_USB_CLK(USB_CLK)`
Макрос проверки аргументов типа [RCC_USBClk_TypeDef](#).
- `#define IS_RCC_USB_FREQ(USB_FREQ)`
Макрос проверки аргументов типа [RCC_USBFreq_TypeDef](#).
- `#define IS_RCC_ADC_CLK(ADC_CLK)`
Макрос проверки аргументов типа [RCC_ADCClk_TypeDef](#).
- `#define IS_RCC_SYS_CLK(SYS_CLK)`
Макрос проверки аргументов типа [RCC_SysClk_TypeDef](#).
- `#define IS_RCC_PERIPH_CLK(PERIPH_CLK)`
Макрос проверки аргументов типа [RCC_PeriphClk_TypeDef](#).
- `#define IS_RCC_PERIPH_RST(PERIPH_RST)`
Макрос проверки аргументов типа [RCC_PeriphRst_TypeDef](#).
- `#define IS_RCC_PLLDIV(PLLDIV) (((PLLDIV) & ((uint32_t)0xFFFFF00)) == ((uint32_t)0x00))`
Макрос проверки значения выходного делителя PLL на попадание в допустимый диапазон.
- `#define IS_RCC_PLL_NR(PLL_NR) (((PLL_NR) <= ((uint32_t)33)) && ((PLL_NR) >= ((uint32_t)2)))`
Макрос проверки значения опорного делителя PLL на попадание в допустимый диапазон.
- `#define IS_RCC_PLL_NF(PLL_NF) (((PLL_NF) <= ((uint32_t)513)) && ((PLL_NF) >= ((uint32_t)2)))`
Макрос проверки значения делителя ОС PLL на попадание в допустимый диапазон.
- `#define IS_RCC_CLK_DIV(CLK_DIV) ((CLK_DIV) < ((uint32_t)64))`
Макрос проверки значения делителя тактового сигнала на попадание в допустимый диапазон.

- `#define IS_RCC_SYS_FREQ(SYS_FREQ) (((SYS_FREQ) < ((uint32_t)200000000)) && ((SYS_FREQ) >= ((uint32_t)1000000)))`

Макрос проверки значения желаемой частоты при автонастройке в допустимый диапазон.

Перечисления

- enum `RCC_PLLRef_TypeDef` { `RCC_PLLRef_XI_OSC`, `RCC_PLLRef_USB_CLK`, `RCC_PLLRef_USB_60MHz`, `RCC_PLLRef_ETH_25MHz` }

Выбор источника опорного сигнала PLL.

- enum `RCC_PLLNO_TypeDef` { `RCC_PLLNO_Disable`, `RCC_PLLNO_Div2`, `RCC_PLLNO_Div4 = 3` }

Выходной делитель NO.

- enum `RCC_UARTClk_TypeDef` { `RCC_UARTClk_SYSCLK`, `RCC_UARTClk_XI_OSC`, `RCC_UARTClk_USB_CLK`, `RCC_UARTClk_USB_60MHz` }

Выбор источника тактирования для UART.

- enum `RCC_SPIClk_TypeDef` { `RCC_SPIClk_SYSCLK`, `RCC_SPIClk_XI_OSC`, `RCC_SPIClk_USB_CLK`, `RCC_SPIClk_USB_60MHz` }

Выбор источника тактирования для SPI.

- enum `RCC_USBClk_TypeDef` { `RCC_USBClk_XI_OSC`, `RCC_USBClk_USB_CLK` }

Выбор источника тактирования для USB.

- enum `RCC_USBFreq_TypeDef` { `RCC_USBFreq_12MHz`, `RCC_USBFreq_24MHz` }

Выбор фиксированной частоты на входе CLK_USB.

- enum `RCC_ADCClk_TypeDef` { `RCC_ADCClk_0`, `RCC_ADCClk_1`, `RCC_ADCClk_2`, `RCC_ADCClk_3`, `RCC_ADCClk_4`, `RCC_ADCClk_5`, `RCC_ADCClk_6`, `RCC_ADCClk_7`, `RCC_ADCClk_8`, `RCC_ADCClk_9`, `RCC_ADCClk_10`, `RCC_ADCClk_11` }

Выбор модуля ADC для настройки его тактового сигнала.

- enum `RCC_SysClk_TypeDef` { `RCC_SysClk_CPE_Sel`, `RCC_SysClk_POR`, `RCC_SysClk_XI_OSC`, `RCC_SysClk_PLL`, `RCC_SysClk_PLLDIV`, `RCC_SysClk_USB60MHz`, `RCC_SysClk_USB_CLK`, `RCC_SysClk_ETH25MHz` }

Выбор источника системной частоты.

- enum `RCC_PeriphClk_TypeDef` { `RCC_PeriphClk_QEP0` = ((uint32_t)(1<<1)), `RCC_PeriphClk_QEP1` = ((uint32_t)(1<<2)), `RCC_PeriphClk_CMP` = ((uint32_t)(1<<9)), `RCC_PeriphClk_PWM0` = ((uint32_t)(1<<10)), `RCC_PeriphClk_PWM1` = ((uint32_t)(1<<11)), `RCC_PeriphClk_PWM2` = ((uint32_t)(1<<12)), `RCC_PeriphClk_PWM3` = ((uint32_t)(1<<13)), `RCC_PeriphClk_PWM4` = ((uint32_t)(1<<14)), `RCC_PeriphClk_PWM5` = ((uint32_t)(1<<15)), `RCC_PeriphClk_PWM6` = ((uint32_t)(1<<16)), `RCC_PeriphClk_PWM7` = ((uint32_t)(1<<17)), `RCC_PeriphClk_PWM8` = ((uint32_t)(1<<18)), `RCC_PeriphClk_WD` = ((uint32_t)(1<<19)), `RCC_PeriphClk_I2C0` = ((uint32_t)(1<<20)), `RCC_PeriphClk_I2C1` = ((uint32_t)(1<<21)), `RCC_PeriphClk_ADC` = ((uint32_t)(1<<24)) }

Управление тактированием периферийных блоков

- enum `RCC_PeriphRst_TypeDef` { `RCC_PeriphRst_WD` = ((uint32_t)(1<<0)), `RCC_PeriphRst_I2C0` = ((uint32_t)(1<<1)), `RCC_PeriphRst_I2C1` = ((uint32_t)(1<<2)), `RCC_PeriphRst_USB` = ((uint32_t)(1<<3)), `RCC_PeriphRst_Timer0` = ((uint32_t)(1<<4)), `RCC_PeriphRst_Timer1` = ((uint32_t)(1<<5)), `RCC_PeriphRst_Timer2` = ((uint32_t)(1<<6)), `RCC_PeriphRst_UART0` = ((uint32_t)(1<<7)), `RCC_PeriphRst_UART1` = ((uint32_t)(1<<8)), `RCC_PeriphRst_UART2` = ((uint32_t)(1<<9)), `RCC_PeriphRst_UART3` = ((uint32_t)(1<<10)), `RCC_PeriphRst_SPI0` =

```

((uint32_t)(1<<11)),
RCC_PeriphRst_SPI1 = ((uint32_t)(1<<12)), RCC_PeriphRst_SPI2 = ((uint32_t)(1<<13)),
RCC_PeriphRst_SPI3 = ((uint32_t)(1<<14)), RCC_PeriphRst_ETH = ((uint32_t)(1<<15)),
RCC_PeriphRst_QEP0 = ((uint32_t)(1<<0)), RCC_PeriphRst_QEP1 = ((uint32_t)(1<<1)),
RCC_PeriphRst_PWM0 = ((uint32_t)(1<<2)), RCC_PeriphRst_PWM1 = ((uint32_t)←
t)(1<<3)),
RCC_PeriphRst_PWM2 = ((uint32_t)(1<<4)), RCC_PeriphRst_PWM3 = ((uint32_t)←
t)(1<<5)), RCC_PeriphRst_PWM4 = ((uint32_t)(1<<6)), RCC_PeriphRst_PWM5 =
((uint32_t)(1<<7)),
RCC_PeriphRst_PWM6 = ((uint32_t)(1<<8)), RCC_PeriphRst_PWM7 = ((uint32_t)←
t)(1<<9)), RCC_PeriphRst_PWM8 = ((uint32_t)(1<<10)), RCC_PeriphRst_CAP0 =
((uint32_t)(1<<11)),
RCC_PeriphRst_CAP1 = ((uint32_t)(1<<12)), RCC_PeriphRst_CAP2 = ((uint32_t)←
t)(1<<13)), RCC_PeriphRst_CAP3 = ((uint32_t)(1<<14)), RCC_PeriphRst_CAP4 =
((uint32_t)(1<<15)),
RCC_PeriphRst_CAP5 = ((uint32_t)(1<<16)), RCC_PeriphRst_CMP = ((uint32_t)(1<<17))
}

```

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_U←
ARTClk)
Настройка источника тактового сигнала для выбранного UART.
- void **RCC_UARTClkDivConfig** (**NT_UART_TypeDef** *UARTx, uint32_t DivVal, **Functional←**
State 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.

8.14.1 Подробное описание

Файл содержит все прототипы функций для RCC (Reset & Clock Control).

Автор

НИИЭТ

- Богдан Колбов (bkolbov), 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"

8.14.2 Макросы

8.14.2.1 #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 203

Используется в `RCC_ADCClkCmd()` и `RCC_ADCClkDivConfig()`.

8.14.2.2 #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 273

Используется в `RCC_PeriphClkCmd()`.

8.14.2.3 #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 99

Используется в `RCC_PLLInit()`.

8.14.2.4 #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 78

Используется в `RCC_PLLAutoConfig()` и `RCC_PLLInit()`.

8.14.2.5 `#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 140

Используется в `RCC_SPIClkSel()`.

8.14.2.6 `#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 236

Используется в `RCC_SysClkSel()`.

8.14.2.7 `#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 119

Используется в `RCC_UARTClkSel()`.

8.14.2.8 `#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 159

Используется в `RCC_USBClkConfig()`.

8.14.2.9 `#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 176

Используется в `RCC_USBClkConfig()`.

8.14.2.10 `#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()`.

8.14.3 Перечисления

8.14.3.1 `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 183

8.14.3.2 `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 249

8.14.3.3 enum RCC_PeriphRst_TypeDef

Управление сбросом периферийных блоков

Элементы перечислений

RCC_PeriphRst_WD	Управление сбросом сторожевого таймера
RCC_PeriphRst_I2C0	Управление сбросом блока I2C 0
RCC_PeriphRst_I2C1	Управление сбросом блока I2C 1
RCC_PeriphRst_USB	Управление сбросом блока USB
RCC_PeriphRst_Timer0	Управление сбросом блока Timer 0
RCC_PeriphRst_Timer1	Управление сбросом блока Timer 1
RCC_PeriphRst_Timer2	Управление сбросом блока Timer 2
RCC_PeriphRst_UART0	Управление сбросом блока UART 0
RCC_PeriphRst_UART1	Управление сбросом блока UART 1
RCC_PeriphRst_UART2	Управление сбросом блока UART 2
RCC_PeriphRst_UART3	Управление сбросом блока UART 3
RCC_PeriphRst_SPI0	Управление сбросом блока SPI 0
RCC_PeriphRst_SPI1	Управление сбросом блока SPI 1
RCC_PeriphRst_SPI2	Управление сбросом блока SPI 2
RCC_PeriphRst_SPI3	Управление сбросом блока SPI 3
RCC_PeriphRst_ETH	Управление сбросом блока Ethernet
RCC_PeriphRst_QEP0	Управление сбросом блока QEP 0
RCC_PeriphRst_QEP1	Управление сбросом блока QEP 1
RCC_PeriphRst_PWM0	Управление сбросом блока PWM 0
RCC_PeriphRst_PWM1	Управление сбросом блока PWM 1
RCC_PeriphRst_PWM2	Управление сбросом блока PWM 2
RCC_PeriphRst_PWM3	Управление сбросом блока PWM 3
RCC_PeriphRst_PWM4	Управление сбросом блока PWM 4
RCC_PeriphRst_PWM5	Управление сбросом блока PWM 5
RCC_PeriphRst_PWM6	Управление сбросом блока PWM 6
RCC_PeriphRst_PWM7	Управление сбросом блока PWM 7

RCC_PeriphRst_PWM8 Управление сбросом блока PWM 8
RCC_PeriphRst_CAP0 Управление сбросом блока CAP 0
RCC_PeriphRst_CAP1 Управление сбросом блока CAP 1
RCC_PeriphRst_CAP2 Управление сбросом блока CAP 2
RCC_PeriphRst_CAP3 Управление сбросом блока CAP 3
RCC_PeriphRst_CAP4 Управление сбросом блока CAP 4
RCC_PeriphRst_CAP5 Управление сбросом блока CAP 5
RCC_PeriphRst_CMP Управление сбросом блока аналогового компаратора

См. определение в файле niietcm4_rcc.h строка 293

8.14.3.4 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 88

8.14.3.5 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 66

8.14.3.6 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 128

8.14.3.7 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 220

8.14.3.8 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 107

8.14.3.9 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 149

8.14.3.10 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 166

8.14.4 Функции

8.14.4.1 `void RCC_ADCClkCmd (RCC_ADCClk_TypeDef RCC_ADCClk, FunctionalState State)`

Включение тактирования ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 773

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.

8.14.4.2 void RCC_ADCClkDivConfig (RCC_ADCClk_TypeDef RCC_ADCClk, uint32_t DivVal, FunctionalState DivState)

Настройка делителя тактового сигнала для выбранного ADC.

Аргументы

RCC_ADCClk	Выбор модуля ADC. Параметр принимает любое значение из RCC_ADCClk_TypeDef .
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 689

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.

8.14.4.3 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_Periph↔ Clk	Выбор периферии. Параметр принимает любое значение из RCC_PeriphClk_↔TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 336

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_RCC_PERIPH_CLK.

8.14.4.4 void RCC_PeriphRstCmd (RCC_PeriphRst_TypeDef RCC_PeriphRst, FunctionalState State)

Вывод из состояния сброса периферийных блоков.

Аргументы

RCC_Periph↔ Rst	Выбор периферийного модуля. Параметр принимает любое значение из RCC_↔PeriphRst_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 863

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_RCC_PERIPH_RST, RCC_PeriphRst_↔ETH, RCC_PeriphRst_I2C0, RCC_PeriphRst_I2C1, RCC_PeriphRst_SPI0, RCC_PeriphRst_SPI1, RCC_PeriphRst_SPI2, RCC_PeriphRst_SPI3, RCC_PeriphRst_Timer0, RCC_PeriphRst_Timer1, RCC_PeriphRst_Timer2, RCC_PeriphRst_UART0, RCC_PeriphRst_UART1, RCC_PeriphRst_↔UART2, RCC_PeriphRst_UART3, RCC_PeriphRst_USB и RCC_PeriphRst_WD.

Используется в UART_DeInit().

8.14.4.5 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 Freq < 50 МГц, то в качестве системной частоты будет использован выход делителя PLL DIV. В остальных случаях используется выход PLL напрямую.

Аргументы

RCC_PLLRef	Выбор источника опорного сигнала PLL. Параметр принимает любое значение из RCC_PLLRef_TypeDef .
SysFreq	Желаемая системная частота в Гц. Параметр принимает любые значения из диапазона 1000000-200000000, кратные 1000000.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 142

Перекрестные ссылки EXT_OSC_VALUE, IS_RCC_PLL_REF, IS_RCC_SYS_FREQ, RCC_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().

8.14.4.6 void RCC_PLLDeInit ()

Устанавливает все регистры PLL значениями по умолчанию.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_rcc.c строка 292

Перекрестные ссылки RCC_PLL_CTRL_Reset_Value, RCC_PLL_NF_Reset_Value, RCC_PLLNR_Reset_Value и RCC_PLL_OD_Reset_Value.

8.14.4.7 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:

$$F_{OUT} = (F_{IN} \times NF) / (NO \times NR),$$

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 RCC_PLLInit_TypeDef , которая содержит конфигурационную информацию.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 256

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().

8.14.4.8 void RCC_PLLPowerDownCmd (FunctionalState State)

Управление режимом PowerDown PLL.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 307

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.14.4.9 void RCC_PLLStructInit (RCC_PLLInit_TypeDef * RCC_PLLInit_Struct)

Заполнение каждого члена структуры RCC_PLLInit_Struct значениями по умолчанию.

Аргументы

RCC_PLL↔ Init_Struct	Указатель на структуру типа RCC_PLLInit_TypeDef , которую необходимо проинициализировать.
-------------------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 278

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.

8.14.4.10 void RCC_SPIClkCmd (NT_SPI_TypeDef * SPIx, FunctionalState State)

Включение тактирования SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
------	--

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

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 642

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_SPI_ALL_PERIPH`.

8.14.4.11 `void RCC_SPIClkDivConfig (NT_SPI_TypeDef * SPIx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 604

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_SPI_ALL_PERIPH`.

8.14.4.12 `void RCC_SPIClkSel (NT_SPI_TypeDef * SPIx, RCC_SPIClk_TypeDef RCC_SPIClk)`

Настройка источника тактового сигнала для выбранного SPI.

Аргументы

SPIx	Выбор модуля SPI, где x лежит в диапазоне 0-3.
RCC_SPIClk	Выбор источника тактирования для SPI. Параметр принимает любое значение из RCC_SPIClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 563

Перекрестные ссылки `IS_RCC_SPI_CLK` и `IS_SPI_ALL_PERIPH`.

8.14.4.13 `void RCC_SysClkDiv2Out (FunctionalState State)`

Включение генерации тактового сигнала с частотой равной половине системной на выводе `H[0]`. Функция использует драйвер GPIO для настройки выхода.

Аргументы

State	Выбор состояния. Параметр принимает любое значение из FunctionalState: <ul style="list-style-type: none"> • ENABLE - переводит H[0] в выход включенной альтернативной функцией 2. • DISABLE - переводит H[0] в состояние по умолчанию.
-------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 106

Перекрестные ссылки GPIO_InitTypeDef::GPIO_AltFunc, GPIO_AltFunc_2, GPIO_Init_TypeDef::GPIO_Dir, GPIO_Dir_Out, GPIO_Init(), GPIO_Init_TypeDef::GPIO_Mode, GPIO_Mode_AltFunc, GPIO_Init_TypeDef::GPIO_Out, GPIO_Out_En, GPIO_Init_TypeDef::GPIO_Pin, GPIO_Pin_0, GPIO_StructInit() и IS_FUNCTIONAL_STATE.

8.14.4.14 OperationStatus RCC_SysClkSel (RCC_SysClk_TypeDef RCC_SysClk)

Выбор источника для системного тактового сигнала.

Аргументы

RCC_SysClk	Выбор источника. Параметр принимает любое значение из RCC_SysClk_TypeDef .
------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 359

Перекрестные ссылки IS_RCC_SYS_CLK и RCC_WaitClkChange().

Используется в RCC_PLLAutoConfig().

8.14.4.15 RCC_SysClk_TypeDef RCC_SysClkStatus ()

Текущий источник системного тактового сигнала.

Возвращаемые значения

Значение	из RCC_SysClk_TypeDef
----------	---------------------------------------

См. определение в файле niietcm4_rcc.c строка 388

8.14.4.16 void RCC_UARTClkCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение тактирования UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 520

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_UART_ALL_PERIPH`.

8.14.4.17 `void RCC_UARTClkDivConfig (NT_UART_TypeDef * UARTx, uint32_t DivVal, FunctionalState DivState)`

Настройка делителя тактового сигнала для выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
DivVal	Значение делителя. Результирующий коэффициент деления вычисляется по формуле $(2 \times (\text{DivVal} + 1))$. Параметр принимает любое значение из диапазона 0-63.
DivState	Выбор состояния делителя. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 482

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_RCC_CLK_DIV` и `IS_UART_ALL_PERIPH`.

8.14.4.18 `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_UARTClk	Выбор источника тактирования для UART. Параметр принимает любое значение из RCC_UARTClk_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 442

Перекрестные ссылки `IS_RCC_UART_CLK` и `IS_UART_ALL_PERIPH`.

8.14.4.19 `void RCC_USBClkCmd (FunctionalState State)`

Включение тактирования USB.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле `niietcm4_rcc.c` строка 419

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.14.4.20 void RCC_USBCLKConfig (RCC_USBCLK_TypeDef RCC_USBCLK,
RCC_USBFreq_TypeDef RCC_USBFreq)

Настройка источника тактового сигнала для USB.

Аргументы

RCC_USBCLK	Выбор источника тактирования. Параметр принимает любое значение из RCC_USBCLK_TypeDef .
RCC_USBFreq	Выбор фиксированной частоты на входе CLK_USB. Параметр принимает любое значение из RCC_USBFreq_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_rcc.c строка 402

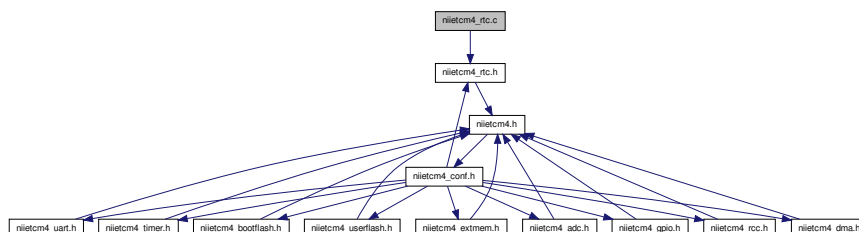
Перекрестные ссылки IS_RCC_USB_CLK и IS_RCC_USB_FREQ.

8.15 Файл niietcm4_rtc.c

Файл содержит реализацию всех функции для работы с RTC.

```
#include "niietcm4_rtc.h"
```

Граф включаемых заголовочных файлов для niietcm4_rtc.c:



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)

8.15.1 Подробное описание

Файл содержит реализацию всех функции для работы с RTC.

Автор

НИИЭТ

- Александр Дыхно (DAV), dykhno@niiet.ru
- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

04.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

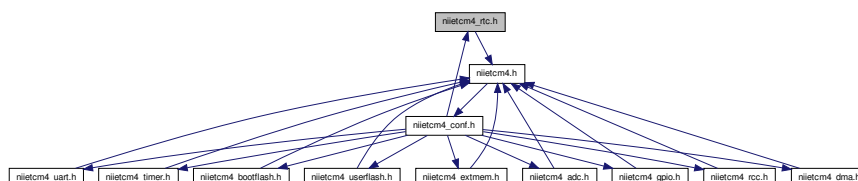
© 2015 ОАО "НИИЭТ"

8.16 Файл niietcm4_rtc.h

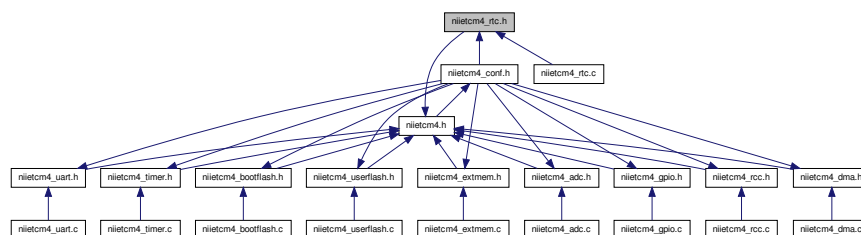
Файл содержит все прототипы функций для таймеров

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_rtc.h:



Граф файлов, в которые включается этот файл:



Структуры данных

- struct `RTC_Time_TypeDef`
Структура времени.
- struct `RTC_Date_TypeDef`
Структура даты.

Макросы

- `#define IS_RTC_PSECOND(PSECOND) ((PSECOND) <= 0x3FF)`
Макрос проверки попадания значений долей секунд в допустимый диапазон.
- `#define IS_RTC_SECOND(SECOND) ((SECOND) <= 59)`
Макрос проверки попадания значений секунд в допустимый диапазон.
- `#define IS_RTC_MINUTE(MINUTE) ((MINUTE) <= 59)`
Макрос проверки попадания значений минут в допустимый диапазон.
- `#define IS_RTC_HOUR(HOUR) ((HOUR) <= 23)`
Макрос проверки попадания значений часов в допустимый диапазон.
- `#define IS_RTC_WEEKDAY(WEEKDAY)`
Макрос проверки аргументов типа `RTC_Weekday_TypeDef`.
- `#define IS_RTC_DAY(DAY) (((DAY) > 0) && ((DAY) <= 31))`
Макрос проверки попадания значений дней в допустимый диапазон.
- `#define IS_RTC_MONTH(MONTH)`
Макрос проверки аргументов типа `RTC_Month_TypeDef`.
- `#define IS_RTC_YEAR(YEAR) ((YEAR) <= 99)`
Макрос проверки попадания значений лет в допустимый диапазон.
- `#define IS_RTC_FORMAT(FORMAT)`
Макрос проверки аргументов типа `RTC_Format_TypeDef`.

Перечисления

- enum `RTC_Weekday_TypeDef` {
`RTC_Weekday_Monday = ((uint32_t)0x01), RTC_Weekday_Tuesday = ((uint32_t)0x02), R←`
`TC_Weekday_Wednesday = ((uint32_t)0x03), RTC_Weekday_Thursday = ((uint32_t)0x04), R←`
`RTC_Weekday_Friday = ((uint32_t)0x05), RTC_Weekday_Saturday = ((uint32_t)0x06), R←`
`TC_Weekday_Sunday = ((uint32_t)0x07) }`
Дни недели.

- enum `RTC_Month_TypeDef` {
`RTC_Month_January` = ((uint32_t)0x01), `RTC_Month_February` = ((uint32_t)0x02), `RTC_Month_March` = ((uint32_t)0x03), `RTC_Month_April` = ((uint32_t)0x04),
`RTC_Month_May` = ((uint32_t)0x05), `RTC_Month_June` = ((uint32_t)0x06), `RTC_Month_July` = ((uint32_t)0x07), `RTC_Month_August` = ((uint32_t)0x08),
`RTC_Month_September` = ((uint32_t)0x09), `RTC_Month_October` = ((uint32_t)0x10), `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`)

8.16.1 Подробное описание

Файл содержит все прототипы функций для таймеров

Автор

НИИЭТ

- Александр Дыхно (DAV), dykhno@niiet.ru
- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

04.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

8.16.2 Макросы

8.16.2.1 #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

8.16.2.2 #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

8.16.2.3 #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

8.16.3 Перечисления

8.16.3.1 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

8.16.3.2 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

8.16.3.3 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
```

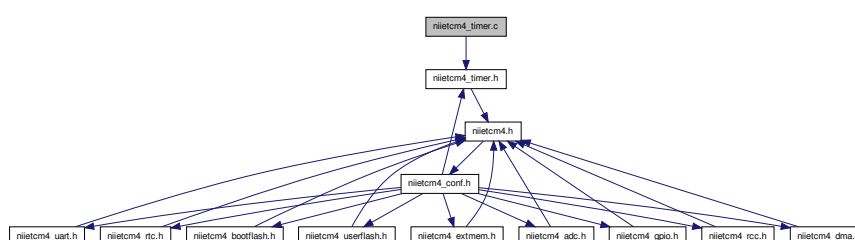
См. определение в файле niietcm4_rtc.h строка 81

8.17 Файл niietcm4_timer.c

Файл содержит реализацию всех функции для работы с таймерами

```
#include "niietcm4_timer.h"
```

Граф включаемых заголовочных файлов для niietcm4_timer.c:



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.

8.17.1 Подробное описание

Файл содержит реализацию всех функций для работы с таймерами

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

03.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.17.2 Функции

8.17.2.1 void `TIMER_Cmd` (NT_TIMER_TypeDef * `TIMERx`, FunctionalState `State`)

Разрешение работы выбранного таймера.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 65Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_TIMER_ALL_PERIPH`.8.17.2.2 void `TIMER_ExtInputConfig` (NT_TIMER_TypeDef * `TIMERx`,
TIMER_ExtInput_TypeDef `TIMER_ExtInput`)

Выбор режима работы входа внешнего тактирования.

Аргументы

<code>TIMERx</code>	Выбор таймера, где x лежит в диапазоне 0-2.
<code>TIMER_ExtInput</code>	Выбор режима работы. Параметр принимает любое значение из TIMER_ExtInput_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 171Перекрестные ссылки `IS_TIMER_ALL_PERIPH`, `IS_TIMER_EXT_INPUT`, `TIMER_ExtInputCountClk` и `TIMER_ExtInputCountEn`.8.17.2.3 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

<code>TIMERx</code>	Выбор таймера, где x лежит в диапазоне 0-2.
<code>TimerClkFreq</code>	Частота в Гц, которой тактируется таймер.
<code>TimerFreq</code>	Частота опустошения таймера в Гц.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 102

Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.17.2.4 uint32_t TIMER_GetCounter (NT_TIMER_TypeDef * TIMEx)

Получение текущего значения счетчика.

Аргументы

TIMEx	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.

8.17.2.5 uint32_t TIMER_GetReload (NT_TIMER_TypeDef * TIMEx)

Получение текущего значения перезагрузки.

Аргументы

TIMEx	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.

8.17.2.6 void TIMER_ITCmd (NT_TIMER_TypeDef * TIMEx, FunctionalState State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMEx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

8.17.2.7 FlagStatus TIMER_ITStatus (NT_TIMER_TypeDef * TIMEx)

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`.

8.17.2.8 `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`.

8.17.2.9 `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`.

8.17.2.10 `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.

8.17.2.11 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

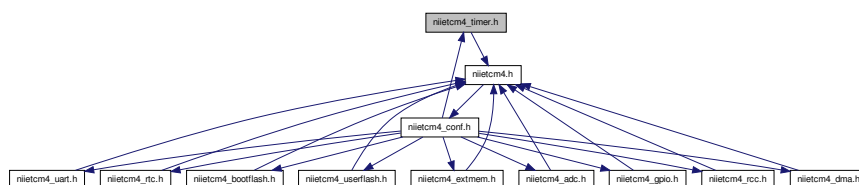
Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.18 Файл niietcm4_timer.h

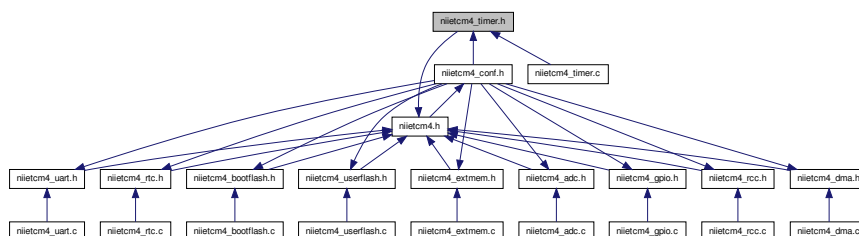
Файл содержит все прототипы функций для таймеров

#include "niietcm4.h"

Граф включаемых заголовочных файлов для niietcm4_timer.h:



Граф файлов, в которые включается этот файл:



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.

8.18.1 Подробное описание

Файл содержит все прототипы функций для таймеров

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.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"

8.18.2 Макросы

8.18.2.1 #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()`.

8.18.3 Перечисления

8.18.3.1 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

8.18.4 Функции

8.18.4.1 void TIMER_Cmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 65

Перекрестные ссылки `IS_FUNCTIONAL_STATE` и `IS_TIMER_ALL_PERIPH`.

```
8.18.4.2 void TIMER_ExtInputConfig ( NT_TIMER_TypeDef * TIMERx,
    TIMER_ExtInput_TypeDef TIMER_ExtInput )
```

Выбор режима работы входа внешнего тактирования.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
TIMER_ExtInput	Выбор режима работы. Параметр принимает любое значение из TIMER_ExtInput_TypeDef .

Возвращаемые значения

Нет

См. определение в файле `niietcm4_timer.c` строка 171

Перекрестные ссылки `IS_TIMER_ALL_PERIPH`, `IS_TIMER_EXT_INPUT`, `TIMER_ExtInputCountClk` и `TIMER_ExtInput_CountEn`.

```
8.18.4.3 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`.

```
8.18.4.4 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.

8.18.4.5 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.

8.18.4.6 void TIMER_ITCmd (NT_TIMER_TypeDef * TIMERx, FunctionalState State)

Разрешение работы прерывания выбранного таймера.

Аргументы

TIMERx	Выбор таймера, где x лежит в диапазоне 0-2.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_timer.c строка 200

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_TIMER_ALL_PERIPH.

8.18.4.7 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.

8.18.4.8 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.

```
8.18.4.9 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.

```
8.18.4.10 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.

```
8.18.4.11 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

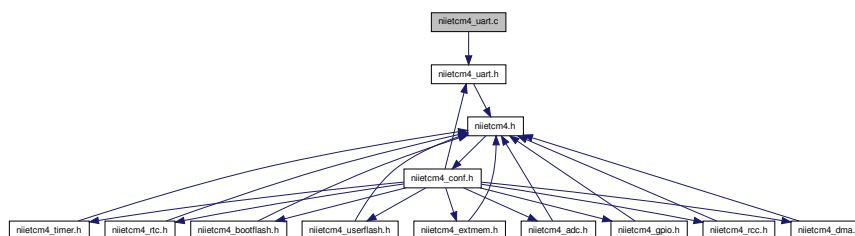
Перекрестные ссылки IS_TIMER_ALL_PERIPH.

8.19 Файл niietcm4_uart.c

Файл содержит реализацию всех функции для работы с модулями UART.

```
#include "niietcm4_uart.h"
```

Граф включаемых заголовочных файлов для niietcm4_uart.c:



Функции

- void **UART_Cmd** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Разрешение работы выбранного UART.
- void **UART_BaudRateDivConfig** (NT_UART_TypeDef *UARTx, uint32_t IntDiv, uint32_t ←
t FracDiv)
Ручная настройка делителя для реализации необходимой скорости передачи.
- void **UART_Break** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Включение разрыва линии.
- void **UART_DeInit** (NT_UART_TypeDef *UARTx)
Устанавливает все регистры UART значениями по умолчанию.
- **OperationStatus** **UART_Init** (NT_UART_TypeDef *UARTx, **UART_Init_TypeDef** *UART ←
_InitStruct)
Инициализирует UARTx согласно параметрам структуры UART_InitStruct.
- void **UART_StructInit** (**UART_Init_TypeDef** *UART_InitStruct)
Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.
- void **UART_SendData** (NT_UART_TypeDef *UARTx, uint32_t Data)
Передача слова данных.
- uint32_t **UART_RecieveData** (NT_UART_TypeDef *UARTx)
Прием слова данных.
- **FlagStatus** **UART_FlagStatus** (NT_UART_TypeDef *UARTx, **UART_Flag_Typedef** UART ←
_Flag)
Запрос состояния выбранного флага.

- `FlagStatus UART_ErrorStatus` (NT_UART_TypeDef *UARTx, `UART_Error_TypeDef` UART_Error)

Запрос состояния выбранного флага ошибки.
- `void UART_ErrorStatusClear` (NT_UART_TypeDef *UARTx)

Очистка флагов ошибки.
- `void UART_ModemConfig` (NT_UART_TypeDef *UARTx, `UART_ModemInit_TypeDef` *UART_ModemInitStruct)

Инициализирует модемный режим UART согласно параметрам структуры `UART_ModemInitStruct`.
- `void UART_ModemStructInit` (`UART_ModemInit_TypeDef` *UART_ModemInitStruct)

Заполнение каждого члена структуры `UART_ModemInitStruct` значениями по умолчанию.
- `void UART_ITFIFOLevelConfig` (NT_UART_TypeDef *UARTx, `UART_Dir_TypeDef` UART_Dir, `UART_FIFOLevel_TypeDef` UART_FIFOLevel)

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.
- `void UART_ITCmd` (NT_UART_TypeDef *UARTx, `UART_ITSource_TypeDef` UART_ITSource, `FunctionalState` State)

Маскирование выбранных прерываний.
- `FlagStatus UART_ITRawStatus` (NT_UART_TypeDef *UARTx, `UART_ITSource_TypeDef` UART_ITSource)

Запрос немаскированного состояния прерывания.
- `FlagStatus UART_ITMaskedStatus` (NT_UART_TypeDef *UARTx, `UART_ITSource_TypeDef` UART_ITSource)

Запрос маскированного состояния прерывания.
- `void UART_ITStatusClear` (NT_UART_TypeDef *UARTx, `UART_ITSource_TypeDef` UART_ITSource)

Сброс флагов состояния выбранных прерываний.
- `void UART_DMABlkOnErrCmd` (NT_UART_TypeDef *UARTx, `FunctionalState` State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.
- `void UART_DMACmd` (NT_UART_TypeDef *UARTx, `UART_Dir_TypeDef` UART_Dir, `FunctionalState` State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

8.19.1 Подробное описание

Файл содержит реализацию всех функции для работы с модулями UART.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niet.ru

Дата

18.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.19.2 Функции

8.19.2.1 void UART_BaudRateDivConfig (NT_UART_TypeDef * UARTx, uint32_t IntDiv, uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

8.19.2.2 void UART_Break (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.19.2.3 void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.19.2.4 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().

8.19.2.5 void UART_DMABlkOnErrCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.19.2.6 void UART_DMACmd (NT_UART_TypeDef * UARTx, UART_Dir_Typedef UART_Dir, FunctionalState State)

Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор направления (прием или передача) для конфигурации. Параметр принимает любое значение из UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DIR и UART_Dir_Rx.

8.19.2.7 FlagStatus UART_ErrorStatus (NT_UART_TypeDef * UARTx, UART_Error_TypeDef UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

8.19.2.8 void UART_ErrorStatusClear (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.19.2.9 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_TypeDef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

8.19.2.10 `OperationStatus UART_Init (NT_UART_TypeDef * UARTx, UART_Init_TypeDef * UART_InitStruct)`

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_Init↔ Struct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле `niietcm4_uart.c` строка 159

Перекрестные ссылки `IS_FUNCTIONAL_STATE`, `IS_UART_ALL_PERIPH`, `IS_UART_DATA_WIDTH`, `IS_UART_FIFO_LEVEL`, `IS_UART_PARITY_BIT`, `IS_UART_STOP_BIT`, `UART_Init_TypeDef::UART_BaudRate`, `UART_Init_TypeDef::UART_ClkFreq`, `UART_Init_TypeDef::UART_DataWidth`, `UART_Init_TypeDef::UART_FIFOEn`, `UART_Init_TypeDef::UART_FIFOLevelRx`, `UART_Init_TypeDef::UART_FIFOLevelTx`, `UART_Init_TypeDef::UART_ParityBit`, `UART_ParityBit_Even`, `UART_ParityBit_High`, `UART_ParityBit_Low`, `UART_ParityBit_Odd`, `UART_Init_TypeDef::UART_RxEn`, `UART_Init_TypeDef::UART_StopBit` и `UART_Init_TypeDef::UART_TxEn`.

8.19.2.11 `void UART_ITCmd (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)`

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 410

Перекрестные ссылки `IS_UART_ALL_PERIPH` и `IS_UART_IT_SOURCE`.

8.19.2.12 `void UART_ITFIFOLevelConfig (NT_UART_TypeDef * UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)`

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор между буфером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFO↔ OLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле `niietcm4_uart.c` строка 384

Перекрестные ссылки `IS_UART_ALL_PERIPH`, `IS_UART_DIR`, `IS_UART_FIFO_LEVEL` и `UART_FIFOLevel_TypeDef`.

ART_Dir_Rx.

8.19.2.13 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.19.2.14 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.19.2.15 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource↔ _TypeDef .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

8.19.2.16 void UART_ModemConfig (NT_UART_TypeDef * UARTx,
UART_ModemInit_TypeDef * UART_ModemInitStruct)

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_Modem↔Init_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit↔_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

8.19.2.17 void UART_ModemStructInit (UART_ModemInit_TypeDef * UART_ModemInitStruct)

Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которую необходимо проинициализировать.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef↔::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::↔UART_RTSEn.

8.19.2.18 uint32_t UART_RecieveData (NT_UART_TypeDef * UARTx)

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.19.2.19 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

8.19.2.20 void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

Аргументы

UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 229

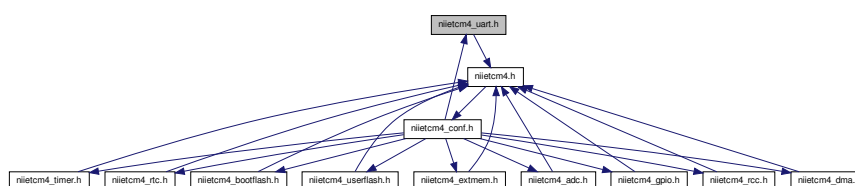
Перекрестные ссылки EXT_OSC_VALUE, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_DataWidth_8, UART_Init_TypeDef::UART_FIFOEn, UART_FIFOLevel_1_2, UART_Init_TypeDef::UART_FIFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Disable, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit, UART_StopBit_1 и UART_Init_TypeDef::UART_TxEn.

8.20 Файл niietcm4_uart.h

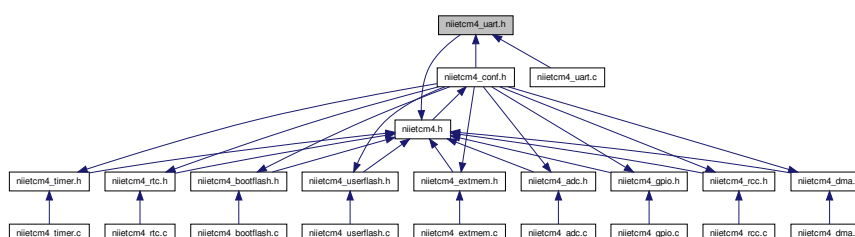
Файл содержит все прототипы функций для UART.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_uart.h:



Граф файлов, в которые включается этот файл:



Структуры данных

- struct `UART_ModemInit_TypeDef`
Структура инициализации модемного режима.
- struct `UART_Init_TypeDef`
Структура инициализации UART.

Макросы

- `#define IS_UART_INT_DIV(INT_DIV) (((INT_DIV) > ((uint32_t)0x0)) && ((INT_DIV) < ((uint32_t)0x10000)))`
Макрос проверки соответствия величины целой части делителя baudrate UART диапазону.
- `#define IS_UART_FRAC_DIV(FRAC_DIV) ((FRAC_DIV) < ((uint32_t)0x40))`
Макрос проверки соответствия величины дробной части делителя baudrate UART диапазону.
- `#define IS_UART_DATA(DATA) ((DATA) < ((uint32_t)0x100))`
Макрос проверки корректности передаваемых данных.
- `#define IS_UART_FLAG(FLAG)`
Макрос проверки аргументов типа `UART_Flag_Typedef`.
- `#define IS_UART_ERROR(ERROR)`
Макрос проверки аргументов типа `UART_Error_Typedef`.
- `#define IS_UART_IT_SOURCE(IT_SOURCE) (((IT_SOURCE) > ((uint32_t)0x0)) && ((IT_SOURCE) < ((uint32_t)0x800)))`
Макрос проверки аргументов типа `UART_ITSource_Typedef`.
- `#define IS_UART_GET_IT_SOURCE(IT_SOURCE)`
Макрос проверки номера пина при работе с пинами по отдельности.
- `#define IS_UART_DIR(DIR)`
Макрос проверки аргументов типа `UART_Dir_Typedef`.
- `#define IS_UART_STOP_BIT(STOP_BIT)`
Макрос проверки аргументов типа `UART_StopBit_TypeDef`.
- `#define IS_UART_PARITY_BIT(PARITY_BIT)`
Макрос проверки аргументов типа `UART_ParityBit_TypeDef`.
- `#define IS_UART_DATA_WIDTH(DATA_WIDTH)`
Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.
- `#define IS_UART_FIFO_LEVEL(FIFO_LEVEL)`
Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

Перечисления

- enum `UART_Flag_Typedef` {
 `UART_Flag_InvCTS`, `UART_Flag_InvDSR`, `UART_Flag_InvDCD`, `UART_Flag_Busy`,
 `UART_Flag_RxFIFOEmpty`, `UART_Flag_TxFIFOFull`, `UART_Flag_RxFIFOFull`, `UART_Flag_TxFIFOEmpty`,
 `UART_Flag_InvRI` }
Перечень флагов.
- enum `UART_Error_Typedef` { `UART_Error_Frame`, `UART_Error_Parity`, `UART_Error_Break`, `UART_Error_Overflow` }
Перечень ошибок приемника.
- enum `UART_ITSource_Typedef` {
 `UART_ITSource_ChangeRI` = ((uint32_t)0x00000001), `UART_ITSource_ChangeCTS` = ((uint32_t)0x00000002), `UART_ITSource_ChangeDCD` = ((uint32_t)0x00000004), `UART_ITSource_ChangeDSR` = ((uint32_t)0x00000008),
 `UART_ITSource_RxFIFOLevel` = ((uint32_t)0x00000010), `UART_ITSource_TxFIFOLevel` =

```
((uint32_t)0x00000020), UART_ITSource_RecieveTimeout = ((uint32_t)0x00000040), UART_ITSource_ErrorFrame = ((uint32_t)0x00000080),
UART_ITSource_ErrorParity = ((uint32_t)0x00000100), UART_ITSource_ErrorBreak = ((uint32_t)0x00000200), UART_ITSource_ErrorOverflow = ((uint32_t)0x00000400) }
```

Источники прерываний UART.

- enum `UART_Dir_TypeDef` { `UART_Dir_Rx`, `UART_Dir_Tx` }

Направления передачи UART.

- enum `UART_StopBit_TypeDef` { `UART_StopBit_1`, `UART_StopBit_2` }

Выбор режима передачи стопового бита.

- enum `UART_ParityBit_TypeDef` { `UART_ParityBit_Disable`, `UART_ParityBit_Odd`, `UART_ParityBit_Even`, `UART_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`, `UART_Flag_TypeDef UART_Flag`)

Запрос состояния выбранного флага.

- `FlagStatus` `UART_ErrorStatus` (`NT_UART_TypeDef *UARTx`, `UART_Error_TypeDef UART_Error`)

Запрос состояния выбранного флага ошибки.

- void `UART_ErrorStatusClear` (`NT_UART_TypeDef *UARTx`)

Очистка флагов ошибки.

- void **UART_ModemConfig** (NT_UART_TypeDef *UARTx, **UART_ModemInit_TypeDef** *UART_ModemInitStruct)
Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInitStruct.
- void **UART_ModemStructInit** (**UART_ModemInit_TypeDef** *UART_ModemInitStruct)
Заполнение каждого члена структуры UART_ModemInitStruct значениями по умолчанию.
- void **UART_ITFIFOLevelConfig** (NT_UART_TypeDef *UARTx, **UART_Dir_Typedef** UART_Dir, **UART_FIFOLevel_TypeDef** UART_FIFOLevel)
Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.
- void **UART_ITCmd** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource, **FunctionalState** State)
Маскирование выбранных прерываний.
- **FlagStatus** **UART_ITRawStatus** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource)
Запрос немаскированного состояния прерывания.
- **FlagStatus** **UART_ITMaskedStatus** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource)
Запрос маскированного состояния прерывания.
- void **UART_ITStatusClear** (NT_UART_TypeDef *UARTx, **UART_ITSource_Typedef** UART_ITSource)
Сброс флагов состояния выбранных прерываний.
- void **UART_DMABlkOnErrCmd** (NT_UART_TypeDef *UARTx, **FunctionalState** State)
Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.
- void **UART_DMACmd** (NT_UART_TypeDef *UARTx, **UART_Dir_Typedef** UART_Dir, **FunctionalState** State)
Разрешение формирования запросов DMA для обслуживания буфера передатчика/приемника

8.20.1 Подробное описание

Файл содержит все прототипы функций для UART.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

18.11.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.20.2 Макросы

8.20.2.1 #define IS_UART_DATA_WIDTH(DATA_WIDTH)

Макроопределение:

```
((DATA_WIDTH) == UART_DataWidth_5) || \
((DATA_WIDTH) == UART_DataWidth_6) || \
((DATA_WIDTH) == UART_DataWidth_7) || \
((DATA_WIDTH) == UART_DataWidth_8))
```

Макрос проверки аргументов типа `UART_DataWidth_TypeDef`.

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

Используется в `UART_Init()`.

8.20.2.2 #define IS_UART_DIR(DIR)

Макроопределение:

```
((DIR) == UART_Dir_Rx) || \
((DIR) == UART_Dir_Tx))
```

Макрос проверки аргументов типа `UART_Dir_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 177

Используется в `UART_DMAMCmd()` и `UART_ITFIFOLevelConfig()`.

8.20.2.3 #define IS_UART_ERROR(ERROR)

Макроопределение:

```
((ERROR) == UART_Error_Frame) || \
((ERROR) == UART_Error_Parity) || \
((ERROR) == UART_Error_Break) || \
((ERROR) == UART_Error_Overflow))
```

Макрос проверки аргументов типа `UART_Error_Typedef`.

См. определение в файле `niietcm4_uart.h` строка 117

Используется в `UART_ErrorStatus()`.

8.20.2.4 #define IS_UART_FIFO_LEVEL(FIFO_LEVEL)

Макроопределение:

```
((FIFO_LEVEL) == UART_FIFOLevel_1_8) || \
((FIFO_LEVEL) == UART_FIFOLevel_1_4) || \
((FIFO_LEVEL) == UART_FIFOLevel_1_2) || \
((FIFO_LEVEL) == UART_FIFOLevel_3_4) || \
((FIFO_LEVEL) == UART_FIFOLevel_7_8))
```

Макрос проверки аргументов типа `UART_FIFOLevel_TypeDef`.

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

Используется в `UART_Init()` и `UART_ITFIFOLevelConfig()`.

8.20.2.5 #define IS_UART_FLAG(FLAG)

Макроопределение:

```
((FLAG) == UART_Flag_InvCTS) || \
  ((FLAG) == UART_Flag_InvDSR) || \
  ((FLAG) == UART_Flag_InvDCD) || \
  ((FLAG) == UART_Flag_Busy) || \
  ((FLAG) == UART_Flag_RxFIFOEmpty) || \
  ((FLAG) == UART_Flag_TxFIFOFull) || \
  ((FLAG) == UART_Flag_RxFIFOFull) || \
  ((FLAG) == UART_Flag_TxFIFOEmpty) || \
  ((FLAG) == UART_Flag_InvRI))
```

Макрос проверки аргументов типа `UART_Flag_Typedef`.

См. определение в файле niietcm4_uart.h строка 91

Используется в `UART_FlagStatus()`.

8.20.2.6 #define IS_UART_GET_IT_SOURCE(IT_SOURCE)

Макроопределение:

```
((IT_SOURCE) == UART_ITSource_ChangeRI) || \
  ((IT_SOURCE) == \
  UART_ITSource_ChangeCTS) || \
  ((IT_SOURCE) == \
  UART_ITSource_ChangeDCD) || \
  ((IT_SOURCE) == \
  UART_ITSource_ChangeDSR) || \
  ((IT_SOURCE) == \
  UART_ITSource_RxFIFOLevel) || \
  ((IT_SOURCE) == \
  UART_ITSource_TxFIFOLevel) || \
  ((IT_SOURCE) == \
  UART_ITSource_RecieveTimeout) || \
  ((IT_SOURCE) == \
  UART_ITSource_ErrorFrame) || \
  ((IT_SOURCE) == \
  UART_ITSource_ErrorParity) || \
  ((IT_SOURCE) == \
  UART_ITSource_ErrorBreak) || \
  ((IT_SOURCE) == \
  UART_ITSource_ErrorOverflow))
```

Макрос проверки номера пина при работе с пинами по отдельности.

См. определение в файле niietcm4_uart.h строка 151

Используется в `UART_ITMaskedStatus()` и `UART_ITRawStatus()`.

8.20.2.7 #define IS_UART_PARITY_BIT(PARITY_BIT)

Макроопределение:

```
((PARITY_BIT) == UART_ParityBit_Disable) || \
  ((PARITY_BIT) == UART_ParityBit_Odd) || \
  ((PARITY_BIT) == UART_ParityBit_Even) || \
  ((PARITY_BIT) == UART_ParityBit_High) || \
  ((PARITY_BIT) == UART_ParityBit_Low))
```

Макрос проверки аргументов типа `UART_ParityBit_TypeDef`.

См. определение в файле niietcm4_uart.h строка 214

Используется в `UART_Init()`.

8.20.2.8 #define IS_UART_STOP_BIT(STOP_BIT)

Макроопределение:

```
((STOP_BIT) == UART_StopBit_1) || \
  ((STOP_BIT) == UART_StopBit_2))
```

Макрос проверки аргументов типа `UART_StopBit_TypeDef`.

См. определение в файле `niietcm4_uart.h` строка 194

Используется в `UART_Init()`.

8.20.3 Перечисления

8.20.3.1 enum UART_DataWidth_TypeDef

Количество передаваемых/принимаемых информационных бит.

Элементы перечислений

`UART_DataWidth_5` Длина информационного слова 5 бит.
`UART_DataWidth_6` Длина информационного слова 6 бит.
`UART_DataWidth_7` Длина информационного слова 7 бит.
`UART_DataWidth_8` Длина информационного слова 8 бит.

См. определение в файле `niietcm4_uart.h` строка 224

8.20.3.2 enum UART_Dir_TypeDef

Направления передачи UART.

Элементы перечислений

`UART_Dir_Rx` Передача.
`UART_Dir_Tx` Прием.

См. определение в файле `niietcm4_uart.h` строка 167

8.20.3.3 enum UART_Error_TypeDef

Перечень ошибок приемника.

Элементы перечислений

`UART_Error_Frame` Флаг ошибки в структуре кадра.
`UART_Error_Parity` Флаг ошибки контроля четности.
`UART_Error_Break` Флаг разрыва линии.
`UART_Error_Overflow` Флаг переполнения буфера приемника.

См. определение в файле `niietcm4_uart.h` строка 105

8.20.3.4 enum UART_FIFOLevel_TypeDef

Порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание

Элементы перечислений

`UART_FIFOLevel_1_8` Заполнение FIFO на 1/8.

UART_FIFOLevel_1_4 Заполнение FIFO на 1/4.
 UART_FIFOLevel_1_2 Заполнение FIFO на 1/2.
 UART_FIFOLevel_3_4 Заполнение FIFO на 3/4.
 UART_FIFOLevel_7_8 Заполнение FIFO на 7/8.

См. определение в файле niietcm4_uart.h строка 246

8.20.3.5 enum UART_Flag_Typedef

Перечень флагов.

Элементы перечислений

UART_Flag_InvCTS Флаг инверсии сигнала на линии UART_CTS.
 UART_Flag_InvDSR Флаг инверсии сигнала на линии UART_DSR.
 UART_Flag_InvDCD Флаг инверсии сигнала на линии UART_DSR.
 UART_Flag_Busy Флаг занятости блока UART.
 UART_Flag_RxFIFOEmpty Флаг пустоты буфера приемника.
 UART_Flag_TxFIFOFull Флаг заполнения буфера передатчика.
 UART_Flag_RxFIFOFull Флаг заполнения буфера приемника.
 UART_Flag_TxFIFOEmpty Флаг пустоты буфера передатчика.
 UART_Flag_InvRI Флаг инверсии сигнала на линии UART_RI.

См. определение в файле niietcm4_uart.h строка 74

8.20.3.6 enum UART_ITSource_Typedef

Источники прерываний UART.

Элементы перечислений

UART_ITSource_ChangeRI Изменение состояния линии UART_RI
 UART_ITSource_ChangeCTS Изменение состояния линии UART_CTS
 UART_ITSource_ChangeDCD Изменение состояния линии UART_DCD
 UART_ITSource_ChangeDSR Изменение состояния линии UART_DSR
 UART_ITSource_RxFIFOLevel Порог переполнения буфера приемника
 UART_ITSource_TxFIFOLevel Порог опустошения буфера передатчика
 UART_ITSource_RecieveTimeout Таймаут приема данных
 UART_ITSource_ErrorFrame Ошибка в структуре кадра
 UART_ITSource_ErrorParity Ошибка контроля четности
 UART_ITSource_ErrorBreak Разрыв линии
 UART_ITSource_ErrorOverflow Переполнение буфера приемника

См. определение в файле niietcm4_uart.h строка 126

8.20.3.7 enum UART_ParityBit_TypeDef

Выбор режима бита четности.

Элементы перечислений

- UART_ParityBit_Disable Не передается, не проверяется.
- UART_ParityBit_Odd Проверка нечетности данных.
- UART_ParityBit_Even Проверка четности данных.
- UART_ParityBit_High Бит четности постоянно равен единице.
- UART_ParityBit_Low Бит четности постоянно равен нулю.

См. определение в файле niietcm4_uart.h строка 201

8.20.3.8 enum UART_StopBit_TypeDef

Выбор режима передачи стопового бита.

Элементы перечислений

- UART_StopBit_1 Один стоповый бит.
- UART_StopBit_2 Два стоповых бита.

См. определение в файле niietcm4_uart.h строка 184

8.20.4 Функции

8.20.4.1 void UART_BaudRateDivConfig (NT_UART_TypeDef * UARTx, uint32_t IntDiv, uint32_t FracDiv)

Ручная настройка делителя для реализации необходимой скорости передачи.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
IntDiv	Целая часть делителя. Параметр принимает любое значение из диапазона 1-65535.
FracDiv	Дробная часть делителя. Параметр принимает любое значение из диапазона 0-63. В случае, если IntDiv равен 65535, значение FracDiv может быть только 0.

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 88

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_FRAC_DIV и IS_UART_INT_DIV.

8.20.4.2 void UART_Break (NT_UART_TypeDef * UARTx, FunctionalState State)

Включение разрыва линии.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 106

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.20.4.3 void UART_Cmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Разрешение работы выбранного UART.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 69

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.20.4.4 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().

8.20.4.5 void UART_DMABlkOnErrCmd (NT_UART_TypeDef * UARTx, FunctionalState State)

Управление блокированием запросов DMA от приемника в случае возникновения прерывания по ошибке.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 502

Перекрестные ссылки IS_FUNCTIONAL_STATE и IS_UART_ALL_PERIPH.

8.20.4.6 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 UART_Dir_Typedef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 520

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DIR и UART_Dir_Rx.

8.20.4.7 FlagStatus UART_ErrorStatus (NT_UART_TypeDef * UARTx, UART_Error_TypeDef UART_Error)

Запрос состояния выбранного флага ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Error	Выбор флага ошибки. Параметр принимает любое значение из UART_Error_TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 305

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_ERROR.

8.20.4.8 void UART_ErrorStatusClear (NT_UART_TypeDef * UARTx)

Очистка флагов ошибки.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 329

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.20.4.9 FlagStatus UART_FlagStatus (NT_UART_TypeDef * UARTx, UART_Flag_TypeDef UART_Flag)

Запрос состояния выбранного флага.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Flag	Выбор флага. Параметр принимает любое значение из UART_Flag_Typedef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 279

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_FLAG.

8.20.4.10 `OperationStatus UART_Init (NT_UART_TypeDef * UARTx, UART_Init_TypeDef * UART_InitStruct)`

Инициализирует UARTx согласно параметрам структуры UART_InitStruct.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3
UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которая содержит конфигурационную информацию.

Возвращаемые значения

Status	Статус результата инициализации. Параметр принимает любое значение из OperationStatus .
--------	---

См. определение в файле niietcm4_uart.c строка 159

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, IS_UART_DATA_WIDTH, IS_UART_FIFO_LEVEL, IS_UART_PARITY_BIT, IS_UART_STOP_BIT, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_Init_TypeDef::UART_FIFOEn, UART_Init_TypeDef::UART_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.

8.20.4.11 `void UART_ITCmd (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef UART_ITSource, FunctionalState State)`

Маскирование выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_ITSource	Выбор прерываний. Параметр принимает любую совокупность значений из UART_ITSource_TypeDef .
State	Выбор состояния. Параметр принимает любое значение из FunctionalState .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 410

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

8.20.4.12 `void UART_ITFIFOLevelConfig (NT_UART_TypeDef * UARTx, UART_Dir_TypeDef UART_Dir, UART_FIFOLevel_TypeDef UART_FIFOLevel)`

Выбор порог заполнения буфера приемника/передатчика, по достижению которого будет генерироваться прерывание.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_Dir	Выбор между буфером приемника и передатчика. Параметр принимает любое из значений UART_Dir_TypeDef .
UART_FIFO← OLevel	Выбор порога. Параметр принимает любое значение из UART_FIFOLevel_← TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 384

Перекрестные ссылки IS_UART_ALL_PERIPH, IS_UART_DIR, IS_UART_FIFO_LEVEL и U←
ART_Dir_Rx.

8.20.4.13 FlagStatus UART_ITMaskedStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос маскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 459

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.20.4.14 FlagStatus UART_ITRawStatus (NT_UART_TypeDef * UARTx,
UART_ITSource_TypeDef UART_ITSource)

Запрос немаскированного состояния прерывания.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT← Source	Выбор прерывания. Параметр принимает любое значение из UART_ITSource← _TypeDef .

Возвращаемые значения

Status	Состояние флага.
--------	------------------

См. определение в файле niietcm4_uart.c строка 433

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_GET_IT_SOURCE.

8.20.4.15 void UART_ITStatusClear (NT_UART_TypeDef * UARTx, UART_ITSource_TypeDef
UART_ITSource)

Сброс флагов состояния выбранных прерываний.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
UART_IT↔ Source	Выбор прерываний. Параметр принимает любое значение из UART_ITSource↔_TypeDef .

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 485

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_IT_SOURCE.

```
8.20.4.16 void UART_ModemConfig ( NT_UART_TypeDef * UARTx,
    UART_ModemInit_TypeDef * UART_ModemInitStruct )
```

Инициализирует модемный режим UART согласно параметрам структуры UART_ModemInit↔Struct.

Аргументы

UART_↔ ModemInit↔ Struct	Указатель на структуру типа UART_ModemInit_TypeDef , которая содержит конфигурационную информацию.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 344

Перекрестные ссылки IS_FUNCTIONAL_STATE, IS_UART_ALL_PERIPH, UART_Modem↔Init_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef::UART_InvDTR, UART_ModemInit↔_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::UART_RTSEn.

```
8.20.4.17 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 UART_ModemInit_TypeDef , которую необходимо проинициализировать.
--------------------------------	--

Возвращаемые значения

Нет

См. определение в файле niietcm4_uart.c строка 365

Перекрестные ссылки UART_ModemInit_TypeDef::UART_CTSEn, UART_ModemInit_TypeDef↔::UART_InvDTR, UART_ModemInit_TypeDef::UART_InvRTS и UART_ModemInit_TypeDef::↔UART_RTSEn.

```
8.20.4.18 uint32_t UART_RecieveData ( NT_UART_TypeDef * UARTx )
```

Прием слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
-------	---

Возвращаемые значения

Data	Слово данных.
------	---------------

См. определение в файле niietcm4_uart.c строка 264

Перекрестные ссылки IS_UART_ALL_PERIPH.

8.20.4.19 void UART_SendData (NT_UART_TypeDef * UARTx, uint32_t Data)

Передача слова данных.

Аргументы

UARTx	Выбор модуля UART, где x лежит в диапазоне 0-3.
Data	Слово данных.

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 250

Перекрестные ссылки IS_UART_ALL_PERIPH и IS_UART_DATA.

8.20.4.20 void UART_StructInit (UART_Init_TypeDef * UART_InitStruct)

Заполнение каждого члена структуры UART_InitStruct значениями по умолчанию.

Аргументы

UART_InitStruct	Указатель на структуру типа UART_Init_TypeDef , которую необходимо проинициализировать.
-----------------	---

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_uart.c строка 229

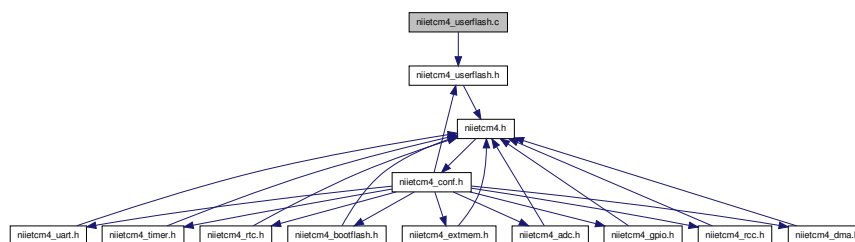
Перекрестные ссылки EXT_OSC_VALUE, UART_Init_TypeDef::UART_BaudRate, UART_Init_TypeDef::UART_ClkFreq, UART_Init_TypeDef::UART_DataWidth, UART_DataWidth_8, UART_Init_TypeDef::UART_FIFOEn, UART_FIFOLevel_1_2, UART_Init_TypeDef::UART_FIFOLevelRx, UART_Init_TypeDef::UART_FIFOLevelTx, UART_Init_TypeDef::UART_ParityBit, UART_ParityBit_Disable, UART_Init_TypeDef::UART_RxEn, UART_Init_TypeDef::UART_StopBit, UART_StopBit_1 и UART_Init_TypeDef::UART_TxEn.

8.21 Файл niietcm4_userflash.c

Файл содержит реализацию всех функции для работы с пользовательской флеш.

```
#include "niietcm4_userflash.h"
```

Граф включаемых заголовочных файлов для niietcm4_userflash.c:



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.

8.21.1 Подробное описание

Файл содержит реализацию всех функции для работы с пользовательской флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

07.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.21.2 Функции

8.21.2.1 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.

8.21.2.2 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.

8.21.2.3 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.

8.21.2.4 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.

8.21.2.5 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

8.21.2.6 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет	
-----	--

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.21.2.7 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флеш.

Возвращаемые значения

Status	Статус работы. Параметр передает любое значение из USERFLASH_↵_Status_TypeDef .
--------	---

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

8.21.2.8 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().

8.21.2.9 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 USERFL↵ASH_PAGE_SIZE_BYTES.

8.21.2.10 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, US↵ERFLASH_OperationStatus(), USERFLASH_OperationStatusClear() и USERFLASH_Status_None.

8.21.2.11 void USERFLASH_Write (uint32_t Address, uint32_t Data)

Запись байта в основную область пользовательской флэш по указанному адресу.

Аргументы

Address	Адрес записи.
Data	Байт данных.

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 148

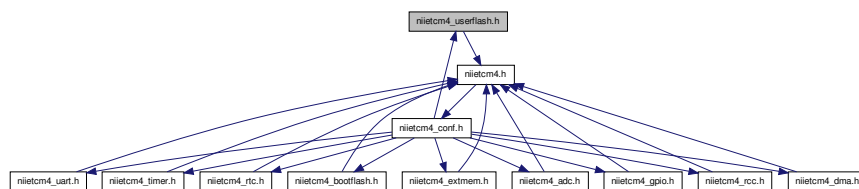
Перекрестные ссылки USERFLASH_MAGIC_KEY.

8.22 Файл niietcm4_userflash.h

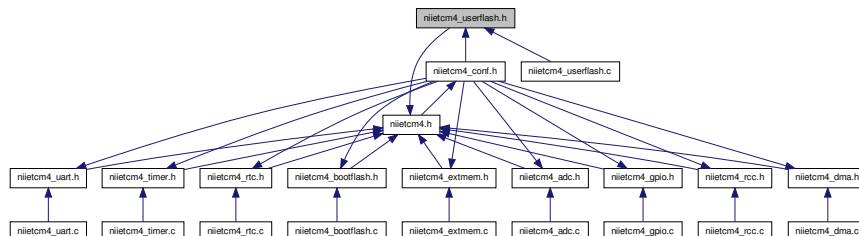
Файл содержит все прототипы функций для пользовательской флеш.

```
#include "niietcm4.h"
```

Граф включаемых заголовочных файлов для niietcm4_userflash.h:



Граф файлов, в которые включается этот файл:



Макросы

- `#define USERFLASH_OPERATION_TIMEOUT ((uint32_t)10000000)`
Время ожидания выполнения операции с флеш.
- `#define USERFLASH_MAGIC_KEY ((uint32_t)0xA4420000)`
Ключ для проведения операций с контроллером пользовательской флеш.
- `#define USERFLASH_PAGE_SIZE_BYTES ((uint32_t)256)`
- `#define USERFLASH_PAGE_TOTAL ((uint32_t)256)`
- `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`
- `#define IS_USERFLASH_PAGE_NUM(PAGE_NUM) (PAGE_NUM < USERFLASH_PAGE_TOTAL)`
Макрос проверки номера страницы основной области пользовательской флеш на попадание в допустимый диапазон.

- `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTES`
- `#define USERFLASH_INFO_PAGE_TOTAL ((uint32_t)2)`
- `#define USERFLASH_INFO_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_INFO_PAGE_TOTAL)`
- `#define IS_USERFLASH_INFO_PAGE_NUM(PAGE_NUM) (PAGE_NUM < USERFLASH_INFO_PAGE_TOTAL)`
Макрос проверки номера страницы информационной области пользовательской флеш на попадание в допустимый диапазон.
- `#define IS_USERFLASH_STATUS(STATUS)`
Макрос проверки аргументов типа `USERFLASH_StatusTypeDef`.

Перечисления

- `enum USERFLASH_StatusTypeDef { USERFLASH_Status_None = ((uint32_t)0), USERFLASH_Status_Complete = ((uint32_t)1), USERFLASH_Status_Error = ((uint32_t)3) }`
Статус работы контроллера пользовательской флеш-памяти.

Функции

- `void USERFLASH_Init (uint32_t SysClkFreq)`
Инициализирует тайминги доступа для контроллера пользовательской флеш.
- `USERFLASH_StatusTypeDef 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.

8.22.1 Подробное описание

Файл содержит все прототипы функций для пользовательской флеш.

Автор

НИИЭТ

- Богдан Колбов (bkolbov), kolbov@niiet.ru

Дата

07.12.2015

Внимание

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДНАЗНАЧЕНО ДЛЯ ОЗНАКОМИТЕЛЬНЫХ ЦЕЛЕЙ И НАПРАВЛЕНО ТОЛЬКО НА ПРЕДОСТАВЛЕНИЕ ДОПОЛНИТЕЛЬНОЙ ИНФОРМАЦИИ О ПРОДУКТЕ, С ЦЕЛЬЮ СОХРАНИТЬ ВРЕМЯ ПОТРЕБИТЕЛЮ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ ИЛИ ПРАВООБЛАДАТЕЛИ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА ПРЯМОЙ ИЛИ КОСВЕННЫЙ УЩЕРБ, ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

© 2015 ОАО "НИИЭТ"

8.22.2 Макросы

8.22.2.1 `#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

8.22.2.2 `#define USERFLASH_INFO_PAGE_SIZE_BYTES USERFLASH_PAGE_SIZE_BYTE↵` `ES`

Размер страницы в байтах.

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

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

Общее количество страниц.

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

8.22.2.4 `#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

8.22.2.5 `#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()`.

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

Общее количество страниц.

См. определение в файле niietcm4_userflash.h строка 69

8.22.2.7 `#define USERFLASH_TOTAL_BYTES (USERFLASH_PAGE_SIZE_BYTES*USERFLASH_PAGE_TOTAL)`

Общий размер основной области.

См. определение в файле niietcm4_userflash.h строка 70

8.22.3 Перечисления

8.22.3.1 `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

8.22.4 Функции

8.22.4.1 `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`.

8.22.4.2 `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.

8.22.4.3 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.

8.22.4.4 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.

8.22.4.5 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

8.22.4.6 void USERFLASH_ITCmd (FunctionalState State)

Включение прерывания по завершению чтения/записи/стирания.

Аргументы

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

Возвращаемые значения

Нет

См. определение в файле niietcm4_userflash.c строка 234

Перекрестные ссылки IS_FUNCTIONAL_STATE.

8.22.4.7 USERFLASH_Status_TypeDef USERFLASH_OperationStatus ()

Статус работы контроллера пользовательской флэш.

Возвращаемые значения

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

См. определение в файле niietcm4_userflash.c строка 79

Используется в USERFLASH_FullErase(), USERFLASH_Info_Read() и USERFLASH_Read().

8.22.4.8 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().

8.22.4.9 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.

8.22.4.10 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`.

8.22.4.11 `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`.

Предметный указатель

- `_CHANNEL_CFG_bits`, 327
 - `CYCLE_CTRL`, 327
 - `DST_INC`, 327
 - `DST_PROT_BUFFERABLE`, 328
 - `DST_PROT_CACHEABLE`, 328
 - `DST_PROT_PRIVILEGED`, 328
 - `DST_SIZE`, 328
 - `N_MINUS_1`, 328
 - `NEXT_USEBURST`, 328
 - `R_POWER`, 328
 - `SRC_INC`, 329
 - `SRC_PROT_BUFFERABLE`, 329
 - `SRC_PROT_CACHEABLE`, 329
 - `SRC_PROT_PRIVILEGED`, 329
 - `SRC_SIZE`, 329
- Битовые операции, 133
 - `GPIO_ClearBits`, 133
 - `GPIO_SetBits`, 133
 - `GPIO_ToggleBits`, 134
- Цифровые компараторы, 46, 53
 - `ADC_DC_DeInit`, 46
 - `ADC_DC_ITCmd`, 53
 - `ADC_DC_ITConfig`, 54
 - `ADC_DC_ITGenCmd`, 54
 - `ADC_DC_ITMaskCmd`, 54
 - `ADC_DC_ITMaskedStatus`, 56
 - `ADC_DC_ITRawStatus`, 56
 - `ADC_DC_ITStatusClear`, 56
 - `ADC_DC_Init`, 46
 - `ADC_DC_StructInit`, 47
- Чтение, 129
 - `GPIO_Read`, 129
 - `GPIO_ReadBit`, 129
 - `GPIO_ReadMask`, 130
- Чтение и запись, 128
- Драйвер периферии, 320
- Фильтрация, 135
 - `GPIO_QualCmd`, 135
 - `GPIO_QualConfig`, 135
 - `GPIO_SyncCmd`, 136
- Функции, 37, 65, 91, 110, 125, 149, 168, 172, 187, 204
 - `ADC_Cmd`, 38
 - `ADC_DC_Cmd`, 38
 - `ADC_DC_GetLastData`, 38
 - `ADC_DC_TrigStatus`, 39
 - `ADC_DC_TrigStatusClear`, 39
 - `ADC_SEQ_Cmd`, 39
 - `ADC_SEQ_FIFOEmptyStatus`, 40
 - `ADC_SEQ_FIFOEmptyStatusClear`, 40
 - `ADC_SEQ_FIFOFullStatus`, 40
 - `ADC_SEQ_FIFOFullStatusClear`, 41
 - `ADC_SEQ_GetConversionCount`, 41
 - `ADC_SEQ_GetFIFOData`, 41
 - `ADC_SEQ_GetFIFOLoad`, 42
 - `ADC_SEQ_SWReq`, 42
 - `BOOTFLASH_ITCmd`, 65
 - `BOOTFLASH_Init`, 65
 - `BOOTFLASH_OperationStatus`, 66
 - `BOOTFLASH_OperationStatusClear`, 66
 - `EXTMEM_DeInit`, 110
 - `EXTMEM_Init`, 110
 - `EXTMEM_StructInit`, 110
 - `RCC_SysClkDiv2Out`, 149
 - `UART_BaudRateDivConfig`, 187
 - `UART_Break`, 188
 - `UART_Cmd`, 188
 - `USERFLASH_ITCmd`, 204
 - `USERFLASH_Init`, 204
 - `USERFLASH_OperationStatus`, 205
 - `USERFLASH_OperationStatusClear`, 205
- Инициализация, 43
- Информационная область флеш, 63, 70, 202, 208
 - `BOOTFLASH_INFO_PAGE_SIZE_BYT↔ES`, 63
 - `BOOTFLASH_INFO_PAGE_TOTAL`, 63
 - `BOOTFLASH_INFO_TOTAL_BYTES`, 63
 - `BOOTFLASH_Info_PageErase`, 70
 - `BOOTFLASH_Info_Write`, 70
 - `USERFLASH_INFO_PAGE_SIZE_BYT↔ES`, 202
 - `USERFLASH_INFO_PAGE_TOTAL`, 202
 - `USERFLASH_INFO_TOTAL_BYTES`, 202
 - `USERFLASH_Info_PageErase`, 208
 - `USERFLASH_Info_Read`, 208
 - `USERFLASH_Info_Write`, 209
- Инициализация и деинициализация, 126, 189
 - `GPIO_DeInit`, 126
 - `GPIO_Init`, 126
 - `GPIO_StructInit`, 127
 - `UART_DeInit`, 189
 - `UART_Init`, 189
 - `UART_StructInit`, 190
- Инициализация каналов DMA, 92
 - `DMA_ChannelDeInit`, 92
 - `DMA_ChannelInit`, 92
 - `DMA_ChannelStructInit`, 93
- Инициализация контроллера DMA, 94

- DMA_DeInit, 94
- DMA_Init, 94
- DMA_StructInit, 95
- Конфигурация, 173
 - TIMER_Cmd, 173
 - TIMER_ExtInputConfig, 174
 - TIMER_FreqConfig, 174
 - TIMER_GetCounter, 174
 - TIMER_GetReload, 175
 - TIMER_PeriodConfig, 175
 - TIMER_SetCounter, 175
 - TIMER_SetReload, 176
- Конфигурация PLL, 151
 - RCC_PLLAutoConfig, 151
 - RCC_PLLDeInit, 152
 - RCC_PLLInit, 152
 - RCC_PLLPowerDownCmd, 153
 - RCC_PLLStructInit, 153
- Конфигурация контроллера DMA, 96
 - DMA_BasePtrConfig, 96
 - DMA_ChannelEnableCmd, 97
 - DMA_HighPriorityCmd, 97
 - DMA_MasterEnableCmd, 97
 - DMA_PrmAltCmd, 98
 - DMA_ProtectionConfig, 98
 - DMA_ReqMaskCmd, 98
 - DMA_SWRequestCmd, 99
 - DMA_UseBurstCmd, 99
- Конфигурация прерываний, 52
- Конфигурация секвенсоров для DMA, 50
 - ADC_SEQ_DMACmd, 50
 - ADC_SEQ_DMAConfig, 50
 - ADC_SEQ_DMAErrorStatus, 51
 - ADC_SEQ_DMAErrorStatusClear, 51
- Константы, 15, 61, 72, 103, 120, 139, 171, 186, 200
 - RCC_CLK_CHANGE_TIMEOUT, 139
- Макросы, 322
- Маски адреса, 104
 - EXTMEM_CEMask_Addr_11, 104
 - EXTMEM_CEMask_Addr_11_19, 104
 - EXTMEM_CEMask_Addr_12, 104
 - EXTMEM_CEMask_Addr_13, 104
 - EXTMEM_CEMask_Addr_14, 105
 - EXTMEM_CEMask_Addr_15, 105
 - EXTMEM_CEMask_Addr_16, 105
 - EXTMEM_CEMask_Addr_17, 105
 - EXTMEM_CEMask_Addr_18, 105
 - EXTMEM_CEMask_Addr_19, 105
- Маски для CHANNEL_CFG, 73
 - CHANNEL_CFG_CYCLE_CTRL_Msk, 73
 - CHANNEL_CFG_CYCLE_CTRL_Pos, 73
 - CHANNEL_CFG_DST_INC_Msk, 73
 - CHANNEL_CFG_DST_INC_Pos, 74
 - CHANNEL_CFG_DST_PROT_CTRL_↔ Msk, 74
 - CHANNEL_CFG_DST_PROT_CTRL_↔ Pos, 74
 - CHANNEL_CFG_DST_SIZE_Msk, 74
 - CHANNEL_CFG_DST_SIZE_Pos, 74
 - CHANNEL_CFG_N_MINUS_1_Msk, 74
 - CHANNEL_CFG_N_MINUS_1_Pos, 74
 - CHANNEL_CFG_NEXT_USEBURST_↔ Msk, 74
 - CHANNEL_CFG_NEXT_USEBURST_↔ Pos, 74
 - CHANNEL_CFG_R_POWER_Msk, 75
 - CHANNEL_CFG_R_POWER_Pos, 75
 - CHANNEL_CFG_SRC_INC_Msk, 75
 - CHANNEL_CFG_SRC_INC_Pos, 75
 - CHANNEL_CFG_SRC_PROT_CTRL_↔ Msk, 75
 - CHANNEL_CFG_SRC_PROT_CTRL_↔ Pos, 75
 - CHANNEL_CFG_SRC_SIZE_Msk, 75
 - CHANNEL_CFG_SRC_SIZE_Pos, 75
- Маски каналов DMA, 76
 - DMA_Channel_All, 76
 - IS_GET_DMA_CHANNEL, 76
- Маски каналов для измерений, 16
 - ADC_Channel_0, 16
 - ADC_Channel_1, 16
 - ADC_Channel_10, 17
 - ADC_Channel_11, 17
 - 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, 18
 - ADC_Channel_2, 18
 - 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, 19
 - ADC_Channel_8, 19
 - ADC_Channel_9, 19
 - ADC_Channel_All, 19
 - ADC_Channel_None, 19
- Маски каналов по имени, 78
 - DMA_Channel_ADCSEQ0, 78
 - DMA_Channel_ADCSEQ1, 78
 - DMA_Channel_ADCSEQ2, 79
 - DMA_Channel_ADCSEQ3, 79
 - DMA_Channel_ADCSEQ4, 79
 - DMA_Channel_ADCSEQ5, 79
 - DMA_Channel_ADCSEQ6, 79
 - DMA_Channel_ADCSEQ7, 79
 - DMA_Channel_SPI0_RX, 79
 - DMA_Channel_SPI0_TX, 79

- DMA_Channel_SPI1_RX, [79](#)
- DMA_Channel_SPI1_TX, [80](#)
- DMA_Channel_SPI2_RX, [80](#)
- DMA_Channel_SPI2_TX, [80](#)
- DMA_Channel_SPI3_RX, [80](#)
- DMA_Channel_SPI3_TX, [80](#)
- DMA_Channel_UART0_RX, [80](#)
- DMA_Channel_UART0_TX, [80](#)
- DMA_Channel_UART1_RX, [80](#)
- DMA_Channel_UART1_TX, [80](#)
- DMA_Channel_UART2_RX, [80](#)
- DMA_Channel_UART2_TX, [81](#)
- DMA_Channel_UART3_RX, [81](#)
- DMA_Channel_UART3_TX, [81](#)
- Маски каналов по номеру, [82](#)
 - DMA_Channel_0, [82](#)
 - DMA_Channel_1, [82](#)
 - DMA_Channel_10, [83](#)
 - DMA_Channel_11, [83](#)
 - DMA_Channel_12, [83](#)
 - DMA_Channel_13, [83](#)
 - DMA_Channel_14, [83](#)
 - DMA_Channel_15, [83](#)
 - DMA_Channel_16, [83](#)
 - DMA_Channel_17, [83](#)
 - DMA_Channel_18, [83](#)
 - DMA_Channel_19, [84](#)
 - DMA_Channel_2, [84](#)
 - DMA_Channel_20, [84](#)
 - DMA_Channel_21, [84](#)
 - DMA_Channel_22, [84](#)
 - DMA_Channel_23, [84](#)
 - DMA_Channel_3, [84](#)
 - DMA_Channel_4, [84](#)
 - DMA_Channel_5, [84](#)
 - DMA_Channel_6, [84](#)
 - DMA_Channel_7, [85](#)
 - DMA_Channel_8, [85](#)
 - DMA_Channel_9, [85](#)
- Маски пинов, [121](#)
 - GPIO_Pin_0, [121](#)
 - GPIO_Pin_0_3, [121](#)
 - GPIO_Pin_0_7, [122](#)
 - GPIO_Pin_1, [122](#)
 - GPIO_Pin_10, [122](#)
 - GPIO_Pin_11, [122](#)
 - GPIO_Pin_12, [122](#)
 - GPIO_Pin_12_15, [122](#)
 - GPIO_Pin_13, [122](#)
 - GPIO_Pin_14, [122](#)
 - GPIO_Pin_15, [122](#)
 - GPIO_Pin_2, [123](#)
 - GPIO_Pin_3, [123](#)
 - GPIO_Pin_4, [123](#)
 - GPIO_Pin_4_7, [123](#)
 - GPIO_Pin_5, [123](#)
 - GPIO_Pin_6, [123](#)
 - GPIO_Pin_7, [123](#)
 - GPIO_Pin_8, [123](#)
 - GPIO_Pin_8_11, [123](#)
 - GPIO_Pin_8_15, [123](#)
 - GPIO_Pin_9, [124](#)
 - GPIO_Pin_All, [124](#)
 - IS_GET_GPIO_PIN, [124](#)
- Маски портов, [268](#)
 - GPIO_Regs_A_C_E_G_Mask, [268](#)
 - GPIO_Regs_B_D_F_H_Mask, [268](#)
 - GPIO_Regs_GPIOA_Mask, [268](#)
 - GPIO_Regs_GPIOB_Mask, [268](#)
 - GPIO_Regs_GPIOC_Mask, [269](#)
 - GPIO_Regs_GPIOD_Mask, [269](#)
 - GPIO_Regs_GPIOE_Mask, [269](#)
 - GPIO_Regs_GPIOF_Mask, [269](#)
 - GPIO_Regs_GPIOG_Mask, [269](#)
 - GPIO_Regs_GPIOH_Mask, [269](#)
- Маски выбора цифровых компараторов, [20](#)
 - ADC_DC_0, [20](#)
 - ADC_DC_1, [20](#)
 - ADC_DC_10, [21](#)
 - ADC_DC_11, [21](#)
 - 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, [22](#)
 - ADC_DC_2, [22](#)
 - 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, [23](#)
 - ADC_DC_8, [23](#)
 - 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, [25](#)
 - ADC_SEQ_6, [25](#)
 - ADC_SEQ_7, [25](#)
- Модули АЦП, [44](#)
 - ADC_DeInit, [44](#)
 - ADC_Init, [44](#)
 - ADC_StructInit, [45](#)

- Начальные значения регистров, 214, 243, 257, 265, 281
- EXT_MEM_CFG_Reset_Value, 257
 - GPIO_DATAOUT_Reset_Value, 265
 - GPIO_GPIODEN0_Reset_Value, 265
 - GPIO_GPIODEN1_Reset_Value, 265
 - GPIO_GPIODEN2_Reset_Value, 265
 - GPIO_GPIODEN3_Reset_Value, 266
 - GPIO_GPIOODCTLx_Reset_Value, 266
 - GPIO_GPIOODSCTLx_Reset_Value, 266
 - GPIO_GPIOPCTLx_Reset_Value, 266
 - GPIO_GPIOPUCTLx_Reset_Value, 266
 - GPIO_GPIOQEx_Reset_Value, 266
 - GPIO_GPIOQMx_Reset_Value, 266
 - GPIO_GPIOQPx_Reset_Value, 266
 - GPIO_GPIOSEx_Reset_Value, 267
 - RCC_PLL_CTRL_Reset_Value, 281
 - RCC_PLL_NF_Reset_Value, 281
 - RCC_PLL_NR_Reset_Value, 281
 - RCC_PLL_OD_Reset_Value, 281
- Настройка DMA, 198
- UART_DMABlkOnErrCmd, 198
 - UART_DMACmd, 198
- Настройка драйвера, 321
- EXT_OSC_VALUE, 321
 - INT_OSC_VALUE, 321
- Основная область флеш, 62, 67, 201, 206
- BOOTFLASH_FullErase, 67
 - BOOTFLASH_PAGE_SIZE_BYTES, 62
 - BOOTFLASH_PAGE_TOTAL, 62
 - BOOTFLASH_PageErase, 67
 - BOOTFLASH_TOTAL_BYTES, 62
 - BOOTFLASH_Write, 67
 - USERFLASH_FullErase, 206
 - USERFLASH_PAGE_SIZE_BYTES, 201
 - USERFLASH_PAGE_TOTAL, 201
 - USERFLASH_PageErase, 206
 - USERFLASH_Read, 207
 - USERFLASH_TOTAL_BYTES, 201
 - USERFLASH_Write, 207
- Периферия, 325
- Прерывания, 137, 177, 195
- GPIO_ITCmd, 137
 - GPIO_ITConfig, 137
 - GPIO_ITStatusClear, 138
 - TIMER_ITCmd, 177
 - TIMER_ITStatus, 177
 - TIMER_ITStatusClear, 178
 - UART_ITCmd, 195
 - UART_ITFIFOLevelConfig, 196
 - UART_ITMaskedStatus, 196
 - UART_ITRawStatus, 196
 - UART_ITStatusClear, 197
- Прием и передача, 191
- UART_ErrorStatus, 191
 - UART_ErrorStatusClear, 191
 - UART_FlagStatus, 192
 - UART_RecieveData, 192
 - UART_SendData, 192
- Приватные данные, 212, 234, 241, 255, 263, 279, 292, 295, 303, 314
- Приватные функции, 215, 236, 244, 258, 270, 282, 293, 297, 305, 316
- ADC_Cmd, 217
 - ADC_DC_Cmd, 217
 - ADC_DC_DeInit, 217
 - ADC_DC_GetLastData, 218
 - ADC_DC_ITCmd, 218
 - ADC_DC_ITConfig, 219
 - ADC_DC_ITGenCmd, 219
 - ADC_DC_ITMaskCmd, 219
 - ADC_DC_ITMaskedStatus, 221
 - ADC_DC_ITRawStatus, 221
 - ADC_DC_ITStatusClear, 221
 - ADC_DC_Init, 218
 - ADC_DC_StructInit, 222
 - ADC_DC_TrigStatus, 222
 - ADC_DC_TrigStatusClear, 222
 - ADC_DeInit, 223
 - ADC_Init, 223
 - ADC_SEQ_Cmd, 223
 - ADC_SEQ_DMACmd, 224
 - ADC_SEQ_DMAConfig, 224
 - ADC_SEQ_DMAErrorStatus, 225
 - ADC_SEQ_DMAErrorStatusClear, 225
 - ADC_SEQ_DeInit, 224
 - ADC_SEQ_FIFOEmptyStatus, 225
 - ADC_SEQ_FIFOEmptyStatusClear, 225
 - ADC_SEQ_FIFOFullStatus, 226
 - ADC_SEQ_FIFOFullStatusClear, 226
 - ADC_SEQ_GetConversionCount, 226
 - ADC_SEQ_GetFIFOData, 227
 - ADC_SEQ_GetFIFOLoad, 227
 - ADC_SEQ_GetITCount, 227
 - ADC_SEQ_ITCmd, 228
 - ADC_SEQ_ITConfig, 228
 - ADC_SEQ_ITCountRst, 229
 - ADC_SEQ_ITMaskedStatus, 229
 - ADC_SEQ_ITRawStatus, 229
 - ADC_SEQ_ITStatusClear, 230
 - ADC_SEQ_Init, 228
 - ADC_SEQ_SWReq, 230
 - ADC_SEQ_StructInit, 230
 - ADC_StructInit, 230
 - BOOTFLASH_FullErase, 236
 - BOOTFLASH_ITCmd, 238
 - BOOTFLASH_Info_PageErase, 236
 - BOOTFLASH_Info_Write, 238
 - BOOTFLASH_Init, 238
 - BOOTFLASH_OperationStatus, 238
 - BOOTFLASH_OperationStatusClear, 239
 - BOOTFLASH_PageErase, 239
 - BOOTFLASH_Write, 239
 - DMA_BasePtrConfig, 245
 - DMA_ChannelDeInit, 245
 - DMA_ChannelEnableCmd, 245

- DMA_ChannelInit, 245
- DMA_ChannelStructInit, 246
- DMA_ClearErrorStatus, 246
- DMA_DeInit, 247
- DMA_ErrorStatus, 247
- DMA_HighPriorityCmd, 247
- DMA_Init, 247
- DMA_MasterEnableCmd, 249
- DMA_MasterEnableStatus, 249
- DMA_PrmAltCmd, 249
- DMA_ProtectionConfig, 250
- DMA_ReqMaskCmd, 250
- DMA_SWRequestCmd, 252
- DMA_StateStatus, 250
- DMA_StructInit, 250
- DMA_UseBurstCmd, 252
- DMA_WaitOnReqStatus, 252
- EXTMEM_DeInit, 258
- EXTMEM_Init, 258
- EXTMEM_StructInit, 258
- GPIO_ClearBits, 271
- GPIO_DeInit, 271
- GPIO_ITCmd, 272
- GPIO_ITConfig, 272
- GPIO_ITStatusClear, 273
- GPIO_Init, 271
- GPIO_QualCmd, 273
- GPIO_QualConfig, 273
- GPIO_Read, 274
- GPIO_ReadBit, 274
- GPIO_ReadMask, 274
- GPIO_SetBits, 275
- GPIO_StructInit, 275
- GPIO_SyncCmd, 275
- GPIO_ToggleBits, 276
- GPIO_Write, 276
- GPIO_WriteBit, 276
- GPIO_WriteMask, 277
- RCC_ADCClkCmd, 283
- RCC_ADCClkDivConfig, 283
- RCC_PLLAutoConfig, 285
- RCC_PLLDeInit, 285
- RCC_PLLInit, 285
- RCC_PLLPowerDownCmd, 286
- RCC_PLLStructInit, 286
- RCC_PeriphClkCmd, 284
- RCC_PeriphRstCmd, 284
- RCC_SPIClkCmd, 287
- RCC_SPIClkDivConfig, 287
- RCC_SPIClkSel, 287
- RCC_SysClkDiv2Out, 288
- RCC_SysClkSel, 288
- RCC_SysClkStatus, 288
- RCC_UARTClkCmd, 289
- RCC_UARTClkDivConfig, 289
- RCC_UARTClkSel, 289
- RCC_USBClkCmd, 289
- RCC_USBClkConfig, 290
- RCC_WaitClkChange, 290
- TIMER_Cmd, 297
- TIMER_ExtInputConfig, 298
- TIMER_FreqConfig, 298
- TIMER_GetCounter, 298
- TIMER_GetReload, 299
- TIMER_ITCmd, 299
- TIMER_ITStatus, 299
- TIMER_ITStatusClear, 299
- TIMER_PeriodConfig, 300
- TIMER_SetCounter, 300
- TIMER_SetReload, 300
- UART_BaudRateDivConfig, 306
- UART_Break, 306
- UART_Cmd, 307
- UART_DMABlkOnErrCmd, 307
- UART_DMACmd, 307
- UART_DeInit, 307
- UART_ErrorStatus, 308
- UART_ErrorStatusClear, 308
- UART_FlagStatus, 308
- UART_ITCmd, 309
- UART_ITFIFOLevelConfig, 309
- UART_ITMaskedStatus, 310
- UART_ITRawStatus, 310
- UART_ITStatusClear, 310
- UART_Init, 309
- UART_ModemConfig, 311
- UART_ModemStructInit, 311
- UART_RecieveData, 311
- UART_SendData, 312
- UART_StructInit, 312
- USERFLASH_FullErase, 316
- USERFLASH_ITCmd, 318
- USERFLASH_Info_PageErase, 317
- USERFLASH_Info_Read, 317
- USERFLASH_Info_Write, 317
- USERFLASH_Init, 317
- USERFLASH_OperationStatus, 318
- USERFLASH_OperationStatusClear, 318
- USERFLASH_PageErase, 318
- USERFLASH_Read, 319
- USERFLASH_Write, 319
- Приватные константы, 213, 235, 242, 256, 264, 280, 296, 304, 315
- Режим модема, 193
 - UART_ModemConfig, 193
 - UART_ModemStructInit, 193
- Секвенсоры, 48, 58
 - ADC_SEQ_DeInit, 48
 - ADC_SEQ_GetITCount, 58
 - ADC_SEQ_ITCmd, 59
 - ADC_SEQ_ITConfig, 59
 - ADC_SEQ_ITCountRst, 59
 - ADC_SEQ_ITMaskedStatus, 60
 - ADC_SEQ_ITRawStatus, 60
 - ADC_SEQ_ITStatusClear, 60
 - ADC_SEQ_Init, 48

- ADC_SEQ_StructInit, 49
- Статусная информация, 100
 - DMA_ClearErrorStatus, 100
 - DMA_ErrorStatus, 100
 - DMA_MasterEnableStatus, 100
 - DMA_StateStatus, 102
 - DMA_WaitOnReqStatus, 102
- Тактирование ADC, 162
 - RCC_ADCClkCmd, 162
 - RCC_ADCClkDivConfig, 162
- Тактирование SPI, 160
 - RCC_SPIClkCmd, 160
 - RCC_SPIClkDivConfig, 160
 - RCC_SPIClkSel, 161
- Тактирование UART, 158
 - RCC_UARTClkCmd, 158
 - RCC_UARTClkDivConfig, 158
 - RCC_UARTClkSel, 159
- Тактирование USB, 156
 - RCC_USBClkCmd, 156
 - RCC_USBClkConfig, 156
- Типы, 26, 64, 86, 106, 113, 140, 165, 169, 179, 203, 323
 - 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, 33
 - ADC_DC_Channel_1, 33
 - ADC_DC_Channel_10, 33
 - ADC_DC_Channel_11, 33
 - ADC_DC_Channel_12, 33
 - ADC_DC_Channel_13, 33
 - ADC_DC_Channel_14, 33
 - 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, 33
 - ADC_DC_Channel_20, 33
 - ADC_DC_Channel_21, 33
 - ADC_DC_Channel_22, 33
 - ADC_DC_Channel_23, 33
 - ADC_DC_Channel_3, 33
 - ADC_DC_Channel_4, 33
 - ADC_DC_Channel_5, 33
 - ADC_DC_Channel_6, 33
 - ADC_DC_Channel_7, 33
 - ADC_DC_Channel_8, 33
 - ADC_DC_Channel_9, 33
 - 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, 34
 - ADC_DC_Mode_Single, 33
 - ADC_DC_Mode_SingleHyst, 34
 - ADC_DC_Mode_TypeDef, 33
 - ADC_DC_Module_0, 34
 - ADC_DC_Module_1, 34
 - 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, 34
 - ADC_DC_Module_20, 34
 - ADC_DC_Module_21, 34
 - ADC_DC_Module_22, 34
 - ADC_DC_Module_23, 34
 - ADC_DC_Module_3, 34
 - ADC_DC_Module_4, 34
 - ADC_DC_Module_5, 34
 - ADC_DC_Module_6, 34
 - ADC_DC_Module_7, 34
 - ADC_DC_Module_8, 34
 - ADC_DC_Module_9, 34
 - ADC_DC_Module_TypeDef, 34
 - ADC_Measure_Diff, 34
 - ADC_Measure_Single, 34
 - ADC_Measure_TypeDef, 34
 - ADC_Mode_Active, 35
 - ADC_Mode_Powerdown, 35
 - ADC_Mode_StandBy, 35
 - 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, 35
 - ADC_Resolution_10bit, 35
 - ADC_Resolution_12bit, 35
 - ADC_Resolution_TypeDef, 35
 - ADC_SEQ_FIFOLevel_1, 35
 - ADC_SEQ_FIFOLevel_16, 36

- ADC_SEQ_FIFOLevel_2, 35
- ADC_SEQ_FIFOLevel_32, 36
- ADC_SEQ_FIFOLevel_4, 36
- ADC_SEQ_FIFOLevel_8, 36
- 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, 36
- 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, 64
- BOOTFLASH_Status_Error, 64
- BOOTFLASH_Status_None, 64
- BOOTFLASH_Status_TypeDef, 64
- Bit_CLEAR, 117
- Bit_SET, 117
- BitAction, 117
- DMA_ArbitrationRate_1, 89
- DMA_ArbitrationRate_1024, 89
- DMA_ArbitrationRate_128, 89
- DMA_ArbitrationRate_16, 89
- DMA_ArbitrationRate_2, 89
- DMA_ArbitrationRate_256, 89
- DMA_ArbitrationRate_32, 89
- DMA_ArbitrationRate_4, 89
- DMA_ArbitrationRate_512, 89
- DMA_ArbitrationRate_64, 89
- DMA_ArbitrationRate_8, 89
- DMA_ArbitrationRate_TypeDef, 89
- DMA_DataInc_16, 89
- DMA_DataInc_32, 89
- DMA_DataInc_8, 89
- DMA_DataInc_Disable, 89
- DMA_DataInc_TypeDef, 89
- DMA_DataSize_16, 89
- DMA_DataSize_32, 89
- DMA_DataSize_8, 89
- DMA_DataSize_TypeDef, 89
- DMA_Mode_AltMemScatGath, 90
- DMA_Mode_AltPeriphScatGath, 90
- DMA_Mode_AutoReq, 90
- DMA_Mode_Basic, 90
- DMA_Mode_Disable, 90
- DMA_Mode_PingPong, 90
- DMA_Mode_PrmMemScatGath, 90
- DMA_Mode_PrmPeriphScatGath, 90
- DMA_Mode_TypeDef, 89
- DMA_State_Done, 90
- DMA_State_Free, 90
- DMA_State_Pause, 90
- DMA_State_PeriphScatGath, 90
- DMA_State_ReadConfigData, 90
- DMA_State_ReadDstDataEndPtr, 90
- DMA_State_ReadSrcData, 90
- DMA_State_ReadSrcDataEndPtr, 90
- DMA_State_TypeDef, 90
- DMA_State_WaitReq, 90
- DMA_State_WriteDstData, 90
- EXTMEM_RWWaitState_1, 108
- EXTMEM_RWWaitState_2, 108
- EXTMEM_RWWaitState_3, 108
- EXTMEM_RWWaitState_4, 108
- EXTMEM_RWWaitState_5, 108
- EXTMEM_RWWaitState_6, 108
- EXTMEM_RWWaitState_7, 108
- EXTMEM_RWWaitState_8, 108
- EXTMEM_RWWaitState_TypeDef, 108
- EXTMEM_ReadWaitState_1, 108
- EXTMEM_ReadWaitState_2, 108
- EXTMEM_ReadWaitState_3, 108
- EXTMEM_ReadWaitState_4, 108
- EXTMEM_ReadWaitState_5, 108
- EXTMEM_ReadWaitState_6, 108
- EXTMEM_ReadWaitState_7, 108
- EXTMEM_ReadWaitState_8, 108
- EXTMEM_ReadWaitState_TypeDef, 108
- EXTMEM_Width_16bit, 109
- EXTMEM_Width_8bit, 109
- EXTMEM_Width_TypeDef, 108
- EXTMEM_WriteWaitState_1, 109
- EXTMEM_WriteWaitState_2, 109
- EXTMEM_WriteWaitState_3, 109
- EXTMEM_WriteWaitState_4, 109
- EXTMEM_WriteWaitState_5, 109
- EXTMEM_WriteWaitState_6, 109
- EXTMEM_WriteWaitState_7, 109
- EXTMEM_WriteWaitState_8, 109
- EXTMEM_WriteWaitState_TypeDef, 109
- GPIO_AltFunc_1, 117
- GPIO_AltFunc_2, 117
- GPIO_AltFunc_3, 117
- GPIO_AltFunc_TypeDef, 117
- GPIO_Dir_In, 117
- GPIO_Dir_Out, 117
- GPIO_Dir_TypeDef, 117
- GPIO_IntPol_Neg, 117
- GPIO_IntPol_Pos, 117
- GPIO_IntPol_TypeDef, 117

- GPIO_IntType_Edge, 117
- GPIO_IntType_Level, 117
- GPIO_IntType_TypeDef, 117
- GPIO_Load_16mA, 118
- GPIO_Load_8mA, 118
- GPIO_Load_TypeDef, 117
- GPIO_Mode_AltFunc, 118
- GPIO_Mode_IO, 118
- GPIO_Mode_TypeDef, 118
- GPIO_Out_Dis, 118
- GPIO_Out_En, 118
- GPIO_Out_TypeDef, 118
- GPIO_OutMode_OD, 118
- GPIO_OutMode_PP, 118
- GPIO_OutMode_TypeDef, 118
- GPIO_PullUp_Dis, 118
- GPIO_PullUp_En, 118
- GPIO_PullUp_TypeDef, 118
- GPIO_Qual_Dis, 119
- GPIO_Qual_En, 119
- GPIO_Qual_TypeDef, 118
- GPIO_QualMode_3sample, 119
- GPIO_QualMode_6sample, 119
- GPIO_QualMode_TypeDef, 119
- GPIO_Sync_Dis, 119
- GPIO_Sync_En, 119
- GPIO_Sync_TypeDef, 119
- 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, 31
- IS_ADC_SEQ_FIFO_LEVEL, 31
- IS_ADC_SEQ_MODULE, 31
- IS_ADC_SEQ_START_EVENT, 32
- IS_BOOTFLASH_STATUS, 64
- IS_DMA_ARBITRATION_RATE, 87
- IS_DMA_DATA_INC, 87
- IS_DMA_DATA_SIZE, 88
- IS_DMA_MODE, 88
- IS_DMA_STATE, 88
- IS_EXTMEM_READ_WAITSTATE, 107
- IS_EXTMEM_RW_WAITSTATE, 107
- IS_EXTMEM_WIDTH, 107
- IS_EXTMEM_WRITE_WAITSTATE, 107
- IS_GPIO_ALL_PERIPH, 323
- IS_GPIO_ALT_FUNC, 114
- IS_GPIO_DIR, 114
- IS_GPIO_INT_POL, 114
- IS_GPIO_INT_TYPE, 115
- IS_GPIO_LOAD, 115
- IS_GPIO_MODE, 115
- IS_GPIO_OUT, 115
- IS_GPIO_OUT_MODE, 115
- IS_GPIO_PULLUP, 116
- IS_GPIO_QUAL, 116
- IS_GPIO_QUAL_MODE, 116
- IS_GPIO_SYNC, 116
- IS_RCC_ADC_CLK, 142
- IS_RCC_PERIPH_CLK, 142
- IS_RCC_PLL_NO, 143
- IS_RCC_PLL_REF, 143
- IS_RCC_SPI_CLK, 143
- IS_RCC_SYS_CLK, 143
- IS_RCC_UART_CLK, 143
- IS_RCC_USB_CLK, 144
- IS_RCC_USB_FREQ, 144
- IS_RTC_FORMAT, 166
- IS_RTC_MONTH, 166
- IS_RTC_WEEKDAY, 166
- IS_SPI_ALL_PERIPH, 324
- IS_TIMER_ALL_PERIPH, 324
- IS_TIMER_EXT_INPUT, 169
- IS_UART_ALL_PERIPH, 324
- IS_UART_DATA_WIDTH, 180
- IS_UART_DIR, 180
- IS_UART_ERROR, 181
- IS_UART_FIFO_LEVEL, 181
- IS_UART_FLAG, 181
- IS_UART_GET_IT_SOURCE, 181
- IS_UART_PARITY_BIT, 182
- IS_UART_STOP_BIT, 182
- IS_USERFLASH_STATUS, 203
- RCC_ADCClk_0, 144
- RCC_ADCClk_1, 144
- RCC_ADCClk_10, 145
- RCC_ADCClk_11, 145
- RCC_ADCClk_2, 144
- RCC_ADCClk_3, 144
- RCC_ADCClk_4, 144
- RCC_ADCClk_5, 144
- RCC_ADCClk_6, 144
- RCC_ADCClk_7, 144
- RCC_ADCClk_8, 144
- RCC_ADCClk_9, 144
- RCC_ADCClk_TypeDef, 144
- RCC_PLLNO_Disable, 146
- RCC_PLLNO_Div2, 146
- RCC_PLLNO_Div4, 146
- RCC_PLLNO_TypeDef, 146
- RCC_PLLRef_ETH_25MHz, 146
- RCC_PLLRef_TypeDef, 146
- RCC_PLLRef_USB_60MHz, 146
- RCC_PLLRef_USB_CLK, 146
- RCC_PLLRef_XI_OSC, 146
- RCC_PeriphClk_ADC, 145
- RCC_PeriphClk_CMP, 145
- RCC_PeriphClk_I2C0, 145
- RCC_PeriphClk_I2C1, 145
- RCC_PeriphClk_PWM0, 145
- RCC_PeriphClk_PWM1, 145
- RCC_PeriphClk_PWM2, 145

- RCC_PeriphClk_PWM3, [145](#)
- RCC_PeriphClk_PWM4, [145](#)
- RCC_PeriphClk_PWM5, [145](#)
- RCC_PeriphClk_PWM6, [145](#)
- RCC_PeriphClk_PWM7, [145](#)
- RCC_PeriphClk_PWM8, [145](#)
- RCC_PeriphClk_QEP0, [145](#)
- RCC_PeriphClk_QEP1, [145](#)
- RCC_PeriphClk_TypeDef, [145](#)
- RCC_PeriphClk_WD, [145](#)
- RCC_PeriphRst_CAP0, [146](#)
- RCC_PeriphRst_CAP1, [146](#)
- RCC_PeriphRst_CAP2, [146](#)
- RCC_PeriphRst_CAP3, [146](#)
- RCC_PeriphRst_CAP4, [146](#)
- RCC_PeriphRst_CAP5, [146](#)
- RCC_PeriphRst_CMP, [146](#)
- RCC_PeriphRst_ETH, [146](#)
- RCC_PeriphRst_I2C0, [145](#)
- RCC_PeriphRst_I2C1, [145](#)
- RCC_PeriphRst_PWM0, [146](#)
- RCC_PeriphRst_PWM1, [146](#)
- RCC_PeriphRst_PWM2, [146](#)
- RCC_PeriphRst_PWM3, [146](#)
- RCC_PeriphRst_PWM4, [146](#)
- RCC_PeriphRst_PWM5, [146](#)
- RCC_PeriphRst_PWM6, [146](#)
- RCC_PeriphRst_PWM7, [146](#)
- RCC_PeriphRst_PWM8, [146](#)
- RCC_PeriphRst_QEP0, [146](#)
- RCC_PeriphRst_QEP1, [146](#)
- RCC_PeriphRst_SPI0, [145](#)
- RCC_PeriphRst_SPI1, [145](#)
- RCC_PeriphRst_SPI2, [146](#)
- RCC_PeriphRst_SPI3, [146](#)
- RCC_PeriphRst_Timer0, [145](#)
- RCC_PeriphRst_Timer1, [145](#)
- RCC_PeriphRst_Timer2, [145](#)
- RCC_PeriphRst_TypeDef, [145](#)
- RCC_PeriphRst_UART0, [145](#)
- RCC_PeriphRst_UART1, [145](#)
- RCC_PeriphRst_UART2, [145](#)
- RCC_PeriphRst_UART3, [145](#)
- RCC_PeriphRst_USB, [145](#)
- RCC_PeriphRst_WD, [145](#)
- RCC_SPIClk_SYSCLK, [147](#)
- RCC_SPIClk_TypeDef, [146](#)
- RCC_SPIClk_USB_60MHz, [147](#)
- RCC_SPIClk_USB_CLK, [147](#)
- RCC_SPIClk_XI_OSC, [147](#)
- RCC_SysClk_CPE_Sel, [147](#)
- RCC_SysClk_ETH25MHz, [147](#)
- RCC_SysClk_PLL, [147](#)
- RCC_SysClk_PLLDIV, [147](#)
- RCC_SysClk_POR, [147](#)
- RCC_SysClk_TypeDef, [147](#)
- RCC_SysClk_USB60MHz, [147](#)
- RCC_SysClk_USB_CLK, [147](#)
- RCC_SysClk_XI_OSC, [147](#)
- RCC_UARTClk_SYSCLK, [147](#)
- RCC_UARTClk_TypeDef, [147](#)
- RCC_UARTClk_USB_60MHz, [147](#)
- RCC_UARTClk_USB_CLK, [147](#)
- RCC_UARTClk_XI_OSC, [147](#)
- RCC_USBClk_TypeDef, [147](#)
- RCC_USBClk_USB_CLK, [147](#)
- RCC_USBClk_XI_OSC, [147](#)
- RCC_USBFreq_12MHz, [148](#)
- RCC_USBFreq_24MHz, [148](#)
- RCC_USBFreq_TypeDef, [147](#)
- RTC_Format_BCD, [167](#)
- RTC_Format_BIN, [167](#)
- RTC_Format_TypeDef, [167](#)
- RTC_Month_April, [167](#)
- RTC_Month_August, [167](#)
- RTC_Month_December, [167](#)
- RTC_Month_February, [167](#)
- RTC_Month_January, [167](#)
- RTC_Month_July, [167](#)
- RTC_Month_June, [167](#)
- RTC_Month_March, [167](#)
- RTC_Month_May, [167](#)
- RTC_Month_November, [167](#)
- RTC_Month_October, [167](#)
- RTC_Month_September, [167](#)
- RTC_Month_TypeDef, [167](#)
- RTC_Weekday_Friday, [167](#)
- RTC_Weekday_Monday, [167](#)
- RTC_Weekday_Saturday, [167](#)
- RTC_Weekday_Sunday, [167](#)
- RTC_Weekday_Thursday, [167](#)
- RTC_Weekday_Tuesday, [167](#)
- RTC_Weekday_TypeDef, [167](#)
- RTC_Weekday_Wednesday, [167](#)
- TIMER_ExtInput_CountClk, [169](#)
- TIMER_ExtInput_CountEn, [169](#)
- TIMER_ExtInput_Disable, [169](#)
- TIMER_ExtInput_TypeDef, [169](#)
- UART_DataWidth_5, [183](#)
- UART_DataWidth_6, [183](#)
- UART_DataWidth_7, [183](#)
- UART_DataWidth_8, [183](#)
- UART_DataWidth_TypeDef, [182](#)
- UART_Dir_Rx, [183](#)
- UART_Dir_Tx, [183](#)
- UART_Dir_Typedef, [183](#)
- UART_Error_Break, [183](#)
- UART_Error_Frame, [183](#)
- UART_Error_Overflow, [183](#)
- UART_Error_Parity, [183](#)
- UART_Error_Typedef, [183](#)
- UART_FIFOLevel_1_2, [183](#)
- UART_FIFOLevel_1_4, [183](#)
- UART_FIFOLevel_1_8, [183](#)
- UART_FIFOLevel_3_4, [183](#)
- UART_FIFOLevel_7_8, [183](#)

- UART_FIFOLevel_TypeDef, 183
- UART_Flag_Busy, 184
- UART_Flag_InvCTS, 184
- UART_Flag_InvDCD, 184
- UART_Flag_InvDSR, 184
- UART_Flag_InvRI, 184
- UART_Flag_RxFIFOEmpty, 184
- UART_Flag_RxFIFOFull, 184
- UART_Flag_TxFIFOEmpty, 184
- UART_Flag_TxFIFOFull, 184
- UART_Flag_Typedef, 183
- UART_ITSource_ChangeCTS, 184
- UART_ITSource_ChangeDCD, 184
- UART_ITSource_ChangeDSR, 184
- UART_ITSource_ChangeRI, 184
- UART_ITSource_ErrorBreak, 184
- UART_ITSource_ErrorFrame, 184
- UART_ITSource_ErrorOverflow, 184
- UART_ITSource_ErrorParity, 184
- UART_ITSource_RecieveTimeout, 184
- UART_ITSource_RxFIFOLevel, 184
- UART_ITSource_TxFIFOLevel, 184
- UART_ITSource_Typedef, 184
- UART_ParityBit_Disable, 184
- UART_ParityBit_Even, 184
- UART_ParityBit_High, 184
- UART_ParityBit_Low, 184
- UART_ParityBit_Odd, 184
- UART_ParityBit_TypeDef, 184
- UART_StopBit_1, 185
- UART_StopBit_2, 185
- UART_StopBit_TypeDef, 184
- USERFLASH_Status_Complete, 203
- USERFLASH_Status_Error, 203
- USERFLASH_Status_None, 203
- USERFLASH_Status_TypeDef, 203
- Управление сбросом, 164
 - RCC_PeriphRstCmd, 164
- Управление тактированием, 154
 - RCC_PeriphClkCmd, 154
 - RCC_SysClkSel, 155
 - RCC_SysClkStatus, 155
- Запись, 131
 - GPIO_Write, 131
 - GPIO_WriteBit, 131
 - GPIO_WriteMask, 132
- ADC, 210
 - ADC_Average
 - ADC_Init_TypeDef, 331
 - ADC_Average_16
 - Типы, 32
 - niietcm4_adc.h, 392
 - ADC_Average_2
 - Типы, 32
 - niietcm4_adc.h, 391
 - ADC_Average_32
 - Типы, 32
 - niietcm4_adc.h, 392
 - ADC_Average_4
 - Типы, 32
 - niietcm4_adc.h, 392
 - ADC_Average_64
 - Типы, 32
 - niietcm4_adc.h, 392
 - ADC_Average_8
 - Типы, 32
 - niietcm4_adc.h, 392
 - ADC_Average_Disable
 - Типы, 32
 - niietcm4_adc.h, 391
 - ADC_Average_TypeDef
 - Типы, 32
 - niietcm4_adc.h, 391
 - ADC_Channel_0
 - Маски каналов для измерений, 16
 - niietcm4_adc.h, 381
 - ADC_Channel_1
 - Маски каналов для измерений, 16
 - niietcm4_adc.h, 381
 - ADC_Channel_10
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 381
 - ADC_Channel_11
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 381
 - ADC_Channel_12
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_13
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_14
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_15
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_16
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_17
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_18
 - Маски каналов для измерений, 17
 - niietcm4_adc.h, 382
 - ADC_Channel_19
 - Маски каналов для измерений, 18
 - niietcm4_adc.h, 382
 - ADC_Channel_2
 - Маски каналов для измерений, 18
 - niietcm4_adc.h, 382
 - ADC_Channel_20
 - Маски каналов для измерений, 18
 - niietcm4_adc.h, 383
 - ADC_Channel_21

- Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_22
 - Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_23
 - Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_3
 - Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_4
 - Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_5
 - Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_6
 - Маски каналов для измерений, [18](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_7
 - Маски каналов для измерений, [19](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_8
 - Маски каналов для измерений, [19](#)
 - [niietcm4_adc.h](#), [383](#)
- ADC_Channel_9
 - Маски каналов для измерений, [19](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_Channel_All
 - Маски каналов для измерений, [19](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_Channel_None
 - Маски каналов для измерений, [19](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_Channels
 - [ADC_SEQ_Init_TypeDef](#), [332](#)
- ADC_Cmd
 - Функции, [38](#)
 - Приватные функции, [217](#)
 - [niietcm4_adc.c](#), [359](#)
 - [niietcm4_adc.h](#), [396](#)
- ADC_DC_0
 - Маски выбора цифровых компараторов, [20](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_1
 - Маски выбора цифровых компараторов, [20](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_10
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_11
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_12
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_13
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_14
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [384](#)
- ADC_DC_15
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_16
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_17
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_18
 - Маски выбора цифровых компараторов, [21](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_19
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_2
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_20
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_21
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [385](#)
- ADC_DC_22
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_23
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_3
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_4
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_5
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_6
 - Маски выбора цифровых компараторов, [22](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_7
 - Маски выбора цифровых компараторов, [23](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_8
 - Маски выбора цифровых компараторов, [23](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_9
 - Маски выбора цифровых компараторов, [23](#)
 - [niietcm4_adc.h](#), [386](#)
- ADC_DC_All
 - Маски выбора цифровых компараторов, [23](#)

- niietcm4_adc.h, 386
- ADC_DC_Channel
 - ADC_DC_Init_TypeDef, 330
- ADC_DC_Channel_0
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_1
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_10
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_11
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_12
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_13
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_14
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_15
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_16
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_17
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_18
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_19
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_2
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_20
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_21
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_22
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_23
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_3
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_4
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_5
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_6
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_7
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_8
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_9
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_None
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Channel_TypeDef
 - Типы, 32
 - niietcm4_adc.h, 392
- ADC_DC_Cmd
 - Функции, 38
 - Приватные функции, 217
 - niietcm4_adc.c, 360
 - niietcm4_adc.h, 396
- ADC_DC_Condition
 - ADC_DC_Init_TypeDef, 330
- ADC_DC_Condition_High
 - Типы, 33
 - niietcm4_adc.h, 393
- ADC_DC_Condition_Low
 - Типы, 33
 - niietcm4_adc.h, 393
- ADC_DC_Condition_TypeDef
 - Типы, 33
 - niietcm4_adc.h, 392
- ADC_DC_Condition_Window
 - Типы, 33
 - niietcm4_adc.h, 393
- ADC_DC_DeInit
 - Цифровые компараторы, 46
 - Приватные функции, 217
 - niietcm4_adc.c, 360
 - niietcm4_adc.h, 396
- ADC_DC_GetLastData
 - Функции, 38
 - Приватные функции, 218
 - niietcm4_adc.c, 360
 - niietcm4_adc.h, 397
- ADC_DC_ITCmd
 - Цифровые компараторы, 53
 - Приватные функции, 218
 - niietcm4_adc.c, 361
 - niietcm4_adc.h, 397
- ADC_DC_ITConfig

- Цифровые компараторы, [54](#)
- Приватные функции, [219](#)
- `niietcm4_adc.c`, [361](#)
- `niietcm4_adc.h`, [398](#)
- `ADC_DC_ITGenCmd`
 - Цифровые компараторы, [54](#)
 - Приватные функции, [219](#)
 - `niietcm4_adc.c`, [362](#)
 - `niietcm4_adc.h`, [398](#)
- `ADC_DC_ITMaskCmd`
 - Цифровые компараторы, [54](#)
 - Приватные функции, [219](#)
 - `niietcm4_adc.c`, [362](#)
 - `niietcm4_adc.h`, [398](#)
- `ADC_DC_ITMaskedStatus`
 - Цифровые компараторы, [56](#)
 - Приватные функции, [221](#)
 - `niietcm4_adc.c`, [362](#)
 - `niietcm4_adc.h`, [400](#)
- `ADC_DC_ITRawStatus`
 - Цифровые компараторы, [56](#)
 - Приватные функции, [221](#)
 - `niietcm4_adc.c`, [363](#)
 - `niietcm4_adc.h`, [400](#)
- `ADC_DC_ITStatusClear`
 - Цифровые компараторы, [56](#)
 - Приватные функции, [221](#)
 - `niietcm4_adc.c`, [363](#)
 - `niietcm4_adc.h`, [400](#)
- `ADC_DC_Init`
 - Цифровые компараторы, [46](#)
 - Приватные функции, [218](#)
 - `niietcm4_adc.c`, [361](#)
 - `niietcm4_adc.h`, [397](#)
- `ADC_DC_Init_TypeDef`, [329](#)
 - `ADC_DC_Channel`, [330](#)
 - `ADC_DC_Condition`, [330](#)
 - `ADC_DC_Mode`, [330](#)
 - `ADC_DC_ThresholdHigh`, [330](#)
 - `ADC_DC_ThresholdLow`, [330](#)
- `ADC_DC_Mode`
 - `ADC_DC_Init_TypeDef`, [330](#)
- `ADC_DC_Mode_Multiple`
 - Типы, [33](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Mode_MultipleHyst`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Mode_Single`
 - Типы, [33](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Mode_SingleHyst`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Mode_TypeDef`
 - Типы, [33](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_0`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_1`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_10`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_11`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_12`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_13`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_14`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_15`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_16`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_17`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_18`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_19`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_DC_Module_2`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_20`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_DC_Module_21`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_DC_Module_22`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_DC_Module_23`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_DC_Module_3`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_4`
 - Типы, [34](#)
 - `niietcm4_adc.h`, [393](#)
- `ADC_DC_Module_5`
 - Типы, [34](#)

- niietcm4_adc.h, [393](#)
- ADC_DC_Module_6
 - Типы, [34](#)
 - niietcm4_adc.h, [393](#)
- ADC_DC_Module_7
 - Типы, [34](#)
 - niietcm4_adc.h, [393](#)
- ADC_DC_Module_8
 - Типы, [34](#)
 - niietcm4_adc.h, [393](#)
- ADC_DC_Module_9
 - Типы, [34](#)
 - niietcm4_adc.h, [393](#)
- ADC_DC_Module_TypeDef
 - Типы, [34](#)
 - niietcm4_adc.h, [393](#)
- ADC_DC_None
 - Маски выбора цифровых компараторов, [23](#)
 - niietcm4_adc.h, [387](#)
- ADC_DC_StructInit
 - Цифровые компараторы, [47](#)
 - Приватные функции, [222](#)
 - niietcm4_adc.c, [363](#)
 - niietcm4_adc.h, [401](#)
- ADC_DC_ThresholdHigh
 - ADC_DC_Init_TypeDef, [330](#)
- ADC_DC_ThresholdLow
 - ADC_DC_Init_TypeDef, [330](#)
- ADC_DC_TrigStatus
 - Функции, [39](#)
 - Приватные функции, [222](#)
 - niietcm4_adc.c, [364](#)
 - niietcm4_adc.h, [401](#)
- ADC_DC_TrigStatusClear
 - Функции, [39](#)
 - Приватные функции, [222](#)
 - niietcm4_adc.c, [364](#)
 - niietcm4_adc.h, [401](#)
- ADC_DeInit
 - Модули АЦП, [44](#)
 - Приватные функции, [223](#)
 - niietcm4_adc.c, [364](#)
 - niietcm4_adc.h, [402](#)
- ADC_Init
 - Модули АЦП, [44](#)
 - Приватные функции, [223](#)
 - niietcm4_adc.c, [364](#)
 - niietcm4_adc.h, [402](#)
- ADC_Init_TypeDef, [331](#)
 - ADC_Average, [331](#)
 - ADC_Measure_A, [331](#)
 - ADC_Measure_B, [331](#)
 - ADC_Mode, [331](#)
 - ADC_Phase, [332](#)
 - ADC_Resolution, [332](#)
- ADC_Measure_A
 - ADC_Init_TypeDef, [331](#)
- ADC_Measure_B
 - ADC_Init_TypeDef, [331](#)
- ADC_Init_TypeDef, [331](#)
 - ADC_Measure_Diff
 - Типы, [34](#)
 - niietcm4_adc.h, [394](#)
 - ADC_Measure_Single
 - Типы, [34](#)
 - niietcm4_adc.h, [394](#)
 - ADC_Measure_TypeDef
 - Типы, [34](#)
 - niietcm4_adc.h, [394](#)
 - ADC_Mode
 - ADC_Init_TypeDef, [331](#)
 - ADC_Mode_Active
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
 - ADC_Mode_Powerdown
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
 - ADC_Mode_StandBy
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
 - ADC_Mode_TypeDef
 - Типы, [34](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_0
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_1
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_10
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_11
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_2
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_3
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_4
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_5
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_6
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_7
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_8
 - Типы, [35](#)
 - niietcm4_adc.h, [394](#)
- ADC_Module_9
 - niietcm4_adc.h, [394](#)

- Типы, [35](#)
- `niietcm4_adc.h`, [394](#)
- `ADC_Module_TypeDef`
 - Типы, [35](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_Phase`
 - `ADC_Init_TypeDef`, [332](#)
- `ADC_Resolution`
 - `ADC_Init_TypeDef`, [332](#)
- `ADC_Resolution_10bit`
 - Типы, [35](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_Resolution_12bit`
 - Типы, [35](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_Resolution_TypeDef`
 - Типы, [35](#)
 - `niietcm4_adc.h`, [394](#)
- `ADC_SEQ_0`
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_1`
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_2`
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_3`
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_4`
 - Маски выбора секвенсоров, [24](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_5`
 - Маски выбора секвенсоров, [25](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_6`
 - Маски выбора секвенсоров, [25](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_7`
 - Маски выбора секвенсоров, [25](#)
 - `niietcm4_adc.h`, [387](#)
- `ADC_SEQ_Cmd`
 - Функции, [39](#)
 - Приватные функции, [223](#)
 - `niietcm4_adc.c`, [366](#)
 - `niietcm4_adc.h`, [402](#)
- `ADC_SEQ_ConversionCount`
 - `ADC_SEQ_Init_TypeDef`, [333](#)
- `ADC_SEQ_ConversionDelay`
 - `ADC_SEQ_Init_TypeDef`, [333](#)
- `ADC_SEQ_DC`
 - `ADC_SEQ_Init_TypeDef`, [333](#)
- `ADC_SEQ_DMACmd`
 - Конфигурация секвенсоров для DMA, [50](#)
 - Приватные функции, [224](#)
 - `niietcm4_adc.c`, [366](#)
 - `niietcm4_adc.h`, [403](#)
- `ADC_SEQ_DMAConfig`
 - Конфигурация секвенсоров для DMA, [50](#)
 - Приватные функции, [224](#)
 - `niietcm4_adc.c`, [367](#)
 - `niietcm4_adc.h`, [403](#)
- `ADC_SEQ_DMAErrorStatus`
 - Конфигурация секвенсоров для DMA, [51](#)
 - Приватные функции, [225](#)
 - `niietcm4_adc.c`, [367](#)
 - `niietcm4_adc.h`, [404](#)
- `ADC_SEQ_DMAErrorStatusClear`
 - Конфигурация секвенсоров для DMA, [51](#)
 - Приватные функции, [225](#)
 - `niietcm4_adc.c`, [367](#)
 - `niietcm4_adc.h`, [404](#)
- `ADC_SEQ_DeInit`
 - Приватные функции, [224](#)
 - Секвенсоры, [48](#)
 - `niietcm4_adc.c`, [366](#)
 - `niietcm4_adc.h`, [403](#)
- `ADC_SEQ_FIFOEmptyStatus`
 - Функции, [40](#)
 - Приватные функции, [225](#)
 - `niietcm4_adc.c`, [368](#)
 - `niietcm4_adc.h`, [404](#)
- `ADC_SEQ_FIFOEmptyStatusClear`
 - Функции, [40](#)
 - Приватные функции, [225](#)
 - `niietcm4_adc.c`, [368](#)
 - `niietcm4_adc.h`, [404](#)
- `ADC_SEQ_FIFOFullStatus`
 - Функции, [40](#)
 - Приватные функции, [226](#)
 - `niietcm4_adc.c`, [368](#)
 - `niietcm4_adc.h`, [406](#)
- `ADC_SEQ_FIFOFullStatusClear`
 - Функции, [41](#)
 - Приватные функции, [226](#)
 - `niietcm4_adc.c`, [369](#)
 - `niietcm4_adc.h`, [406](#)
- `ADC_SEQ_FIFOLevel_1`
 - Типы, [35](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_SEQ_FIFOLevel_16`
 - Типы, [36](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_SEQ_FIFOLevel_2`
 - Типы, [35](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_SEQ_FIFOLevel_32`
 - Типы, [36](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_SEQ_FIFOLevel_4`
 - Типы, [36](#)
 - `niietcm4_adc.h`, [395](#)
- `ADC_SEQ_FIFOLevel_8`
 - Типы, [36](#)
 - `niietcm4_adc.h`, [395](#)

- ADC_SEQ_FIFOLevel_TypeDef
 - Типы, [35](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_GetConversionCount
 - Функции, [41](#)
 - Приватные функции, [226](#)
 - niietcm4_adc.c, [369](#)
 - niietcm4_adc.h, [406](#)
- ADC_SEQ_GetFIFOData
 - Функции, [41](#)
 - Приватные функции, [227](#)
 - niietcm4_adc.c, [369](#)
 - niietcm4_adc.h, [407](#)
- ADC_SEQ_GetFIFOLoad
 - Функции, [42](#)
 - Приватные функции, [227](#)
 - niietcm4_adc.c, [369](#)
 - niietcm4_adc.h, [407](#)
- ADC_SEQ_GetITCount
 - Приватные функции, [227](#)
 - Секвенсоры, [58](#)
 - niietcm4_adc.c, [371](#)
 - niietcm4_adc.h, [407](#)
- ADC_SEQ_ITCmd
 - Приватные функции, [228](#)
 - Секвенсоры, [59](#)
 - niietcm4_adc.c, [371](#)
 - niietcm4_adc.h, [409](#)
- ADC_SEQ_ITConfig
 - Приватные функции, [228](#)
 - Секвенсоры, [59](#)
 - niietcm4_adc.c, [372](#)
 - niietcm4_adc.h, [409](#)
- ADC_SEQ_ITCountRst
 - Приватные функции, [229](#)
 - Секвенсоры, [59](#)
 - niietcm4_adc.c, [372](#)
 - niietcm4_adc.h, [409](#)
- ADC_SEQ_ITMaskedStatus
 - Приватные функции, [229](#)
 - Секвенсоры, [60](#)
 - niietcm4_adc.c, [372](#)
 - niietcm4_adc.h, [410](#)
- ADC_SEQ_ITRawStatus
 - Приватные функции, [229](#)
 - Секвенсоры, [60](#)
 - niietcm4_adc.c, [373](#)
 - niietcm4_adc.h, [410](#)
- ADC_SEQ_ITStatusClear
 - Приватные функции, [230](#)
 - Секвенсоры, [60](#)
 - niietcm4_adc.c, [373](#)
 - niietcm4_adc.h, [410](#)
- ADC_SEQ_Init
 - Приватные функции, [228](#)
 - Секвенсоры, [48](#)
 - niietcm4_adc.c, [371](#)
 - niietcm4_adc.h, [407](#)
- ADC_SEQ_Init_TypeDef, [332](#)
 - ADC_Channels, [332](#)
 - ADC_SEQ_ConversionCount, [333](#)
 - ADC_SEQ_ConversionDelay, [333](#)
 - ADC_SEQ_DC, [333](#)
 - ADC_SEQ_SWReqEn, [333](#)
 - ADC_SEQ_StartEvent, [333](#)
- ADC_SEQ_Module_0
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_1
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_2
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_3
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_4
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_5
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_6
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_7
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_Module_TypeDef
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_SWReq
 - Функции, [42](#)
 - Приватные функции, [230](#)
 - niietcm4_adc.c, [374](#)
 - niietcm4_adc.h, [411](#)
- ADC_SEQ_SWReqEn
 - ADC_SEQ_Init_TypeDef, [333](#)
- ADC_SEQ_StartEvent
 - ADC_SEQ_Init_TypeDef, [333](#)
- ADC_SEQ_StartEvent_CMP0
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_StartEvent_CMP1
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_StartEvent_CMP2
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_Cycle
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_ITGPIO
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)

- ADC_SEQ_StartEvent_PWM0
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_PWM1
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_PWM2
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_PWM3
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_PWM4
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_PWM5
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_SWReq
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_StartEvent_TIM
 - Типы, [36](#)
 - niietcm4_adc.h, [396](#)
- ADC_SEQ_StartEvent_TypeDef
 - Типы, [36](#)
 - niietcm4_adc.h, [395](#)
- ADC_SEQ_StructInit
 - Приватные функции, [230](#)
 - Секвенсоры, [49](#)
 - niietcm4_adc.c, [373](#)
 - niietcm4_adc.h, [411](#)
- ADC_StructInit
 - Модули АЦП, [45](#)
 - Приватные функции, [230](#)
 - niietcm4_adc.c, [374](#)
 - niietcm4_adc.h, [411](#)
- ALT_DATA
 - DMA_ConfigData_TypeDef, [338](#)
- BOOTFLASH, [233](#)
- BOOTFLASH_FullErase
 - Основная область флеш, [67](#)
 - Приватные функции, [236](#)
 - niietcm4_bootflash.c, [413](#)
 - niietcm4_bootflash.h, [419](#)
- BOOTFLASH_INFO_PAGE_SIZE_BYTES
 - Информационная область флеш, [63](#)
 - niietcm4_bootflash.h, [417](#)
- BOOTFLASH_INFO_PAGE_TOTAL
 - Информационная область флеш, [63](#)
 - niietcm4_bootflash.h, [417](#)
- BOOTFLASH_INFO_TOTAL_BYTES
 - Информационная область флеш, [63](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_ITCmd
 - Функции, [65](#)
 - Приватные функции, [238](#)
 - niietcm4_bootflash.c, [414](#)
- niietcm4_bootflash.h, [420](#)
- BOOTFLASH_Info_PageErase
 - Информационная область флеш, [70](#)
 - Приватные функции, [236](#)
 - niietcm4_bootflash.c, [413](#)
 - niietcm4_bootflash.h, [419](#)
- BOOTFLASH_Info_Write
 - Информационная область флеш, [70](#)
 - Приватные функции, [238](#)
 - niietcm4_bootflash.c, [413](#)
 - niietcm4_bootflash.h, [419](#)
- BOOTFLASH_Init
 - Функции, [65](#)
 - Приватные функции, [238](#)
 - niietcm4_bootflash.c, [414](#)
 - niietcm4_bootflash.h, [419](#)
- BOOTFLASH_OperationStatus
 - Функции, [66](#)
 - Приватные функции, [238](#)
 - niietcm4_bootflash.c, [414](#)
 - niietcm4_bootflash.h, [420](#)
- BOOTFLASH_OperationStatusClear
 - Функции, [66](#)
 - Приватные функции, [239](#)
 - niietcm4_bootflash.c, [414](#)
 - niietcm4_bootflash.h, [420](#)
- BOOTFLASH_PAGE_SIZE_BYTES
 - Основная область флеш, [62](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_PAGE_TOTAL
 - Основная область флеш, [62](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_PageErase
 - Основная область флеш, [67](#)
 - Приватные функции, [239](#)
 - niietcm4_bootflash.c, [415](#)
 - niietcm4_bootflash.h, [420](#)
- BOOTFLASH_Status_Complete
 - Типы, [64](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_Status_Error
 - Типы, [64](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_Status_None
 - Типы, [64](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_Status_TypeDef
 - Типы, [64](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_TOTAL_BYTES
 - Основная область флеш, [62](#)
 - niietcm4_bootflash.h, [418](#)
- BOOTFLASH_Write
 - Основная область флеш, [67](#)
 - Приватные функции, [239](#)
 - niietcm4_bootflash.c, [415](#)
 - niietcm4_bootflash.h, [420](#)
- BUFFERABLE

- DMA_Protect_TypeDef, [342](#)
- Bit_CLEAR
 - Типы, [117](#)
 - niietcm4_gpio.h, [484](#)
- Bit_SET
 - Типы, [117](#)
 - niietcm4_gpio.h, [484](#)
- BitAction
 - Типы, [117](#)
 - niietcm4_gpio.h, [484](#)
- CACHEABLE
 - DMA_Protect_TypeDef, [342](#)
- CEMask
 - EXTMEM_Init_TypeDef, [343](#)
- CH
 - DMA_ConfigStruct_TypeDef, [340](#)
- CHANNEL_CFG
 - DMA_Channel_TypeDef, [334](#)
- CHANNEL_CFG_CYCLE_CTRL_Msk
 - Маски для CHANNEL_CFG, [73](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_CYCLE_CTRL_Pos
 - Маски для CHANNEL_CFG, [73](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_DST_INC_Msk
 - Маски для CHANNEL_CFG, [73](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_DST_INC_Pos
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_DST_PROT_CTRL_Msk
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_DST_PROT_CTRL_Pos
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_DST_SIZE_Msk
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [436](#)
- CHANNEL_CFG_DST_SIZE_Pos
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_N_MINUS_1_Msk
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_N_MINUS_1_Pos
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_NEXT_USEBURST_Msk
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_NEXT_USEBURST_Pos
 - Маски для CHANNEL_CFG, [74](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_R_POWER_Msk
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_R_POWER_Pos
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_SRC_INC_Msk
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_SRC_INC_Pos
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [437](#)
- CHANNEL_CFG_SRC_PROT_CTRL_Msk
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [438](#)
- CHANNEL_CFG_SRC_PROT_CTRL_Pos
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [438](#)
- CHANNEL_CFG_SRC_SIZE_Msk
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [438](#)
- CHANNEL_CFG_SRC_SIZE_Pos
 - Маски для CHANNEL_CFG, [75](#)
 - niietcm4_dma.h, [438](#)
- CHANNEL_CFG_bit
 - DMA_Channel_TypeDef, [334](#)
- CYCLE_CTRL
 - _CHANNEL_CFG_bits, [327](#)
- DMA, [240](#)
- DMA_ArbitrationRate
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_ArbitrationRate_1
 - Типы, [89](#)
 - niietcm4_dma.h, [445](#)
- DMA_ArbitrationRate_1024
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_128
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_16
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_2
 - Типы, [89](#)
 - niietcm4_dma.h, [445](#)
- DMA_ArbitrationRate_256
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_32
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_4
 - Типы, [89](#)
 - niietcm4_dma.h, [445](#)
- DMA_ArbitrationRate_512
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_64
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_ArbitrationRate_8

- Типы, 89
- niietcm4_dma.h, 445
- DMA_ArbitrationRate_TypeDef
 - Типы, 89
 - niietcm4_dma.h, 445
- DMA_BasePtrConfig
 - Конфигурация контроллера DMA, 96
 - Приватные функции, 245
 - niietcm4_dma.c, 425
 - niietcm4_dma.h, 447
- DMA_Channel
 - DMA_Init_TypeDef, 341
- DMA_Channel_0
 - Маски каналов по номеру, 82
 - niietcm4_dma.h, 438
- DMA_Channel_1
 - Маски каналов по номеру, 82
 - niietcm4_dma.h, 438
- DMA_Channel_10
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 438
- DMA_Channel_11
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 438
- DMA_Channel_12
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 438
- DMA_Channel_13
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 439
- DMA_Channel_14
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 439
- DMA_Channel_15
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 439
- DMA_Channel_16
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 439
- DMA_Channel_17
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 439
- DMA_Channel_18
 - Маски каналов по номеру, 83
 - niietcm4_dma.h, 439
- DMA_Channel_19
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 439
- DMA_Channel_2
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 439
- DMA_Channel_20
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 439
- DMA_Channel_21
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 439
- DMA_Channel_22
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 440
- DMA_Channel_23
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 440
- DMA_Channel_3
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 440
- DMA_Channel_4
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 440
- DMA_Channel_5
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 440
- DMA_Channel_6
 - Маски каналов по номеру, 84
 - niietcm4_dma.h, 440
- DMA_Channel_7
 - Маски каналов по номеру, 85
 - niietcm4_dma.h, 440
- DMA_Channel_8
 - Маски каналов по номеру, 85
 - niietcm4_dma.h, 440
- DMA_Channel_9
 - Маски каналов по номеру, 85
 - niietcm4_dma.h, 440
- DMA_Channel_ADCSEQ0
 - Маски каналов по имени, 78
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ1
 - Маски каналов по имени, 78
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ2
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ3
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ4
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ5
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ6
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_ADCSEQ7
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_All
 - Маски каналов DMA, 76
 - niietcm4_dma.h, 441
- DMA_Channel_SPI0_RX
 - Маски каналов по имени, 79
 - niietcm4_dma.h, 441
- DMA_Channel_SPI0_TX
 - Маски каналов по имени, 79

- niietcm4_dma.h, [442](#)
- DMA_Channel_SPI1_RX
 - Маски каналов по имени, [79](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_SPI1_TX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_SPI2_RX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_SPI2_TX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_SPI3_RX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_SPI3_TX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_TypeDef, [333](#)
 - CHANNEL_CFG, [334](#)
 - CHANNEL_CFG_bit, [334](#)
 - DST_DATA_END, [334](#)
 - SRC_DATA_END, [335](#)
- DMA_Channel_UART0_RX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_UART0_TX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [442](#)
- DMA_Channel_UART1_RX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [443](#)
- DMA_Channel_UART1_TX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [443](#)
- DMA_Channel_UART2_RX
 - Маски каналов по имени, [80](#)
 - niietcm4_dma.h, [443](#)
- DMA_Channel_UART2_TX
 - Маски каналов по имени, [81](#)
 - niietcm4_dma.h, [443](#)
- DMA_Channel_UART3_RX
 - Маски каналов по имени, [81](#)
 - niietcm4_dma.h, [443](#)
- DMA_Channel_UART3_TX
 - Маски каналов по имени, [81](#)
 - niietcm4_dma.h, [443](#)
- DMA_ChannelDeInit
 - Инициализация каналов DMA, [92](#)
 - Приватные функции, [245](#)
 - niietcm4_dma.c, [425](#)
 - niietcm4_dma.h, [447](#)
- DMA_ChannelEnable
 - DMA_Init_TypeDef, [341](#)
- DMA_ChannelEnableCmd
 - Конфигурация контроллера DMA, [97](#)
 - Приватные функции, [245](#)
 - niietcm4_dma.c, [425](#)
 - niietcm4_dma.h, [447](#)
- DMA_ChannelInit
 - Инициализация каналов DMA, [92](#)
 - Приватные функции, [245](#)
 - niietcm4_dma.c, [426](#)
 - niietcm4_dma.h, [448](#)
- DMA_ChannelInit_TypeDef, [335](#)
 - DMA_ArbitrationRate, [336](#)
 - DMA_DstDataEndPtr, [336](#)
 - DMA_DstDataInc, [336](#)
 - DMA_DstDataSize, [336](#)
 - DMA_DstProtect, [336](#)
 - DMA_Mode, [336](#)
 - DMA_NextUseburst, [336](#)
 - DMA_SrcDataEndPtr, [337](#)
 - DMA_SrcDataInc, [337](#)
 - DMA_SrcDataSize, [337](#)
 - DMA_SrcProtect, [337](#)
 - DMA_TransfersTotal, [337](#)
- DMA_ChannelStructInit
 - Инициализация каналов DMA, [93](#)
 - Приватные функции, [246](#)
 - niietcm4_dma.c, [426](#)
 - niietcm4_dma.h, [448](#)
- DMA_ClearErrorStatus
 - Приватные функции, [246](#)
 - Статусная информация, [100](#)
 - niietcm4_dma.c, [427](#)
 - niietcm4_dma.h, [449](#)
- DMA_ConfigData_TypeDef, [337](#)
 - ALT_DATA, [338](#)
 - PRM_DATA, [338](#)
 - RESERVED0, [339](#)
 - RESERVED1, [339](#)
- DMA_ConfigStruct_TypeDef, [339](#)
 - CH, [340](#)
- DMA_DataInc_16
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataInc_32
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataInc_8
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataInc_Disable
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataInc_TypeDef
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataSize_16
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataSize_32
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)

- DMA_DataSize_8
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DataSize_TypeDef
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_DeInit
 - Инициализация контроллера DMA, [94](#)
 - Приватные функции, [247](#)
 - niietcm4_dma.c, [427](#)
 - niietcm4_dma.h, [449](#)
- DMA_DstDataEndPtr
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_DstDataInc
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_DstDataSize
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_DstProtect
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_ErrorStatus
 - Приватные функции, [247](#)
 - Статусная информация, [100](#)
 - niietcm4_dma.c, [427](#)
 - niietcm4_dma.h, [449](#)
- DMA_HighPriority
 - DMA_Init_TypeDef, [341](#)
- DMA_HighPriorityCmd
 - Конфигурация контроллера DMA, [97](#)
 - Приватные функции, [247](#)
 - niietcm4_dma.c, [427](#)
 - niietcm4_dma.h, [449](#)
- DMA_Init
 - Инициализация контроллера DMA, [94](#)
 - Приватные функции, [247](#)
 - niietcm4_dma.c, [428](#)
 - niietcm4_dma.h, [450](#)
- DMA_Init_TypeDef, [340](#)
 - DMA_Channel, [341](#)
 - DMA_ChannelEnable, [341](#)
 - DMA_HighPriority, [341](#)
 - DMA_PrmAlt, [341](#)
 - DMA_Protection, [341](#)
 - DMA_ReqMask, [341](#)
 - DMA_UseBurst, [341](#)
- DMA_MasterEnableCmd
 - Конфигурация контроллера DMA, [97](#)
 - Приватные функции, [249](#)
 - niietcm4_dma.c, [428](#)
 - niietcm4_dma.h, [450](#)
- DMA_MasterEnableStatus
 - Приватные функции, [249](#)
 - Статусная информация, [100](#)
 - niietcm4_dma.c, [428](#)
 - niietcm4_dma.h, [450](#)
- DMA_Mode
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_Mode_AltMemScatGath
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_AltPeriphScatGath
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_AutoReq
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_Basic
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_Disable
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_PingPong
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_PrmMemScatGath
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_PrmPeriphScatGath
 - Типы, [90](#)
 - niietcm4_dma.h, [446](#)
- DMA_Mode_TypeDef
 - Типы, [89](#)
 - niietcm4_dma.h, [446](#)
- DMA_NextUseburst
 - DMA_ChannelInit_TypeDef, [336](#)
- DMA_PrmAlt
 - DMA_Init_TypeDef, [341](#)
- DMA_PrmAltCmd
 - Конфигурация контроллера DMA, [98](#)
 - Приватные функции, [249](#)
 - niietcm4_dma.c, [429](#)
 - niietcm4_dma.h, [451](#)
- DMA_Protect_TypeDef, [342](#)
 - BUFFERABLE, [342](#)
 - CACHEABLE, [342](#)
 - PRIVELGED, [342](#)
- DMA_Protection
 - DMA_Init_TypeDef, [341](#)
- DMA_ProtectionConfig
 - Конфигурация контроллера DMA, [98](#)
 - Приватные функции, [250](#)
 - niietcm4_dma.c, [429](#)
 - niietcm4_dma.h, [451](#)
- DMA_ReqMask
 - DMA_Init_TypeDef, [341](#)
- DMA_ReqMaskCmd
 - Конфигурация контроллера DMA, [98](#)
 - Приватные функции, [250](#)
 - niietcm4_dma.c, [429](#)
 - niietcm4_dma.h, [451](#)
- DMA_SWRequestCmd
 - Конфигурация контроллера DMA, [99](#)
 - Приватные функции, [252](#)
 - niietcm4_dma.c, [430](#)
 - niietcm4_dma.h, [452](#)
- DMA_SrcDataEndPtr

- DMA_ChannelInit_TypeDef, 337
- DMA_SrcDataInc
 - DMA_ChannelInit_TypeDef, 337
- DMA_SrcDataSize
 - DMA_ChannelInit_TypeDef, 337
- DMA_SrcProtect
 - DMA_ChannelInit_TypeDef, 337
- DMA_State_Done
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_Free
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_Pause
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_PeriphScatGath
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_ReadConfigData
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_ReadDstDataEndPtr
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_ReadSrcData
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_ReadSrcDataEndPtr
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_TypeDef
 - Типы, 90
 - niietcm4_dma.h, 446
- DMA_State_WaitReq
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_State_WriteDstData
 - Типы, 90
 - niietcm4_dma.h, 447
- DMA_StateStatus
 - Приватные функции, 250
 - Статусная информация, 102
 - niietcm4_dma.c, 430
 - niietcm4_dma.h, 452
- DMA_StructInit
 - Инициализация контроллера DMA, 95
 - Приватные функции, 250
 - niietcm4_dma.c, 430
 - niietcm4_dma.h, 452
- DMA_TransfersTotal
 - DMA_ChannelInit_TypeDef, 337
- DMA_UseBurst
 - DMA_Init_TypeDef, 341
- DMA_UseBurstCmd
 - Конфигурация контроллера DMA, 99
 - Приватные функции, 252
 - niietcm4_dma.c, 430
 - niietcm4_dma.h, 452
- DMA_WaitOnReqStatus
 - Приватные функции, 252
 - Статусная информация, 102
 - niietcm4_dma.c, 431
 - niietcm4_dma.h, 453
- DST_DATA_END
 - DMA_Channel_TypeDef, 334
- DST_INC
 - _CHANNEL_CFG_bits, 327
- DST_PROT_BUFFERABLE
 - _CHANNEL_CFG_bits, 328
- DST_PROT_CACHEABLE
 - _CHANNEL_CFG_bits, 328
- DST_PROT_PRIVILEGED
 - _CHANNEL_CFG_bits, 328
- DST_SIZE
 - _CHANNEL_CFG_bits, 328
- EXT_MEM_CFG_Reset_Value
 - Начальные значения регистров, 257
 - niietcm4_extmem.c, 454
- EXT_OSC_VALUE
 - Настройка драйвера, 321
 - niietcm4.h, 355
- EXTMEM, 254
- EXTMEM_CEMask_Addr_11
 - Маски адреса, 104
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_11_19
 - Маски адреса, 104
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_12
 - Маски адреса, 104
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_13
 - Маски адреса, 104
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_14
 - Маски адреса, 105
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_15
 - Маски адреса, 105
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_16
 - Маски адреса, 105
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_17
 - Маски адреса, 105
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_18
 - Маски адреса, 105
 - niietcm4_extmem.h, 458
- EXTMEM_CEMask_Addr_19
 - Маски адреса, 105
 - niietcm4_extmem.h, 459
- EXTMEM_DeInit
 - Функции, 110
 - Приватные функции, 258

- niietcm4_extmem.c, [454](#)
- niietcm4_extmem.h, [461](#)
- EXTMEM_Init
 - Функции, [110](#)
 - Приватные функции, [258](#)
 - niietcm4_extmem.c, [455](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_Init_TypeDef, [343](#)
 - CEMask, [343](#)
 - EXTMEM_RWWaitState, [343](#)
 - EXTMEM_ReadWaitState, [343](#)
 - EXTMEM_Width, [343](#)
 - EXTMEM_WriteWaitState, [343](#)
- EXTMEM_RWWaitState
 - EXTMEM_Init_TypeDef, [343](#)
- EXTMEM_RWWaitState_1
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_2
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_3
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_4
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_5
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_6
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_7
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_8
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_RWWaitState_TypeDef
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState
 - EXTMEM_Init_TypeDef, [343](#)
- EXTMEM_ReadWaitState_1
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_2
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_3
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_4
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_5
 - Типы, [108](#)
- niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_6
 - Типы, [108](#)
- EXTMEM_ReadWaitState_7
 - Типы, [108](#)
- EXTMEM_ReadWaitState_8
 - Типы, [108](#)
- EXTMEM_ReadWaitState_TypeDef
 - Типы, [108](#)
- EXTMEM_WriteWaitState
 - EXTMEM_Init_TypeDef, [343](#)
- EXTMEM_WriteWaitState_1
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_2
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_3
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_4
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_5
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_6
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_7
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_8
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_TypeDef
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_6
 - Типы, [108](#)
- niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_7
 - Типы, [108](#)
- niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_8
 - Типы, [108](#)
- niietcm4_extmem.h, [460](#)
- EXTMEM_ReadWaitState_TypeDef
 - Типы, [108](#)
- niietcm4_extmem.h, [460](#)
- EXTMEM_StructInit
 - Функции, [110](#)
 - Приватные функции, [258](#)
 - niietcm4_extmem.c, [455](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_Width
 - EXTMEM_Init_TypeDef, [343](#)
- EXTMEM_Width_16bit
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_Width_8bit
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_Width_TypeDef
 - Типы, [108](#)
 - niietcm4_extmem.h, [460](#)
- EXTMEM_WriteWaitState
 - EXTMEM_Init_TypeDef, [343](#)
- EXTMEM_WriteWaitState_1
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_2
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_3
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_4
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_5
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_6
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_7
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_8
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)
- EXTMEM_WriteWaitState_TypeDef
 - Типы, [109](#)
 - niietcm4_extmem.h, [461](#)

- GPIO, 261
- GPIO_AltFunc
 - GPIO_Init_TypeDef, 344
- GPIO_AltFunc_1
 - Типы, 117
 - niietcm4_gpio.h, 484
- GPIO_AltFunc_2
 - Типы, 117
 - niietcm4_gpio.h, 484
- GPIO_AltFunc_3
 - Типы, 117
 - niietcm4_gpio.h, 484
- GPIO_AltFunc_TypeDef
 - Типы, 117
 - niietcm4_gpio.h, 484
- GPIO_ClearBits
 - Битовые операции, 133
 - Приватные функции, 271
 - niietcm4_gpio.c, 468
 - niietcm4_gpio.h, 487
- GPIO_DATAOUT_Reset_Value
 - Начальные значения регистров, 265
 - niietcm4_gpio.c, 465
- GPIO_DeInit
 - Инициализация и деинициализация, 126
 - Приватные функции, 271
 - niietcm4_gpio.c, 468
 - niietcm4_gpio.h, 487
- GPIO_Dir
 - GPIO_Init_TypeDef, 344
- GPIO_Dir_In
 - Типы, 117
 - niietcm4_gpio.h, 485
- GPIO_Dir_Out
 - Типы, 117
 - niietcm4_gpio.h, 485
- GPIO_Dir_TypeDef
 - Типы, 117
 - niietcm4_gpio.h, 484
- GPIO_GPIODEN0_Reset_Value
 - Начальные значения регистров, 265
 - niietcm4_gpio.c, 465
- GPIO_GPIODEN1_Reset_Value
 - Начальные значения регистров, 265
 - niietcm4_gpio.c, 465
- GPIO_GPIODEN2_Reset_Value
 - Начальные значения регистров, 265
 - niietcm4_gpio.c, 465
- GPIO_GPIODEN3_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIODCTLx_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIODSCTLx_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIOPCTLx_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIOQEx_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIOQMx_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIOQPx_Reset_Value
 - Начальные значения регистров, 266
 - niietcm4_gpio.c, 466
- GPIO_GPIOSEx_Reset_Value
 - Начальные значения регистров, 267
 - niietcm4_gpio.c, 467
- GPIO_ITCmd
 - Прерывания, 137
 - Приватные функции, 272
 - niietcm4_gpio.c, 470
 - niietcm4_gpio.h, 488
- GPIO_ITConfig
 - Прерывания, 137
 - Приватные функции, 272
 - niietcm4_gpio.c, 470
 - niietcm4_gpio.h, 488
- GPIO_ITStatusClear
 - Прерывания, 138
 - Приватные функции, 273
 - niietcm4_gpio.c, 471
 - niietcm4_gpio.h, 489
- GPIO_Init
 - Инициализация и деинициализация, 126
 - Приватные функции, 271
 - niietcm4_gpio.c, 468
 - niietcm4_gpio.h, 487
- GPIO_Init_TypeDef, 344
 - GPIO_AltFunc, 344
 - GPIO_Dir, 344
 - GPIO_Load, 344
 - GPIO_Mode, 345
 - GPIO_Out, 345
 - GPIO_OutMode, 345
 - GPIO_Pin, 345
 - GPIO_PullUp, 345
- GPIO_IntPol_Neg
 - Типы, 117
 - niietcm4_gpio.h, 485
- GPIO_IntPol_Pos
 - Типы, 117
 - niietcm4_gpio.h, 485
- GPIO_IntPol_TypeDef
 - Типы, 117
 - niietcm4_gpio.h, 485
- GPIO_IntType_Edge
 - Типы, 117
 - niietcm4_gpio.h, 485

- GPIO_IntType_Level
 - Типы, [117](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_IntType_TypeDef
 - Типы, [117](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Load
 - GPIO_Init_TypeDef, [344](#)
- GPIO_Load_16mA
 - Типы, [118](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Load_8mA
 - Типы, [118](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Load_TypeDef
 - Типы, [117](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Mode
 - GPIO_Init_TypeDef, [345](#)
- GPIO_Mode_AltFunc
 - Типы, [118](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Mode_IO
 - Типы, [118](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Mode_TypeDef
 - Типы, [118](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_Out
 - GPIO_Init_TypeDef, [345](#)
- GPIO_Out_Dis
 - Типы, [118](#)
 - niietcm4_gpio.h, [486](#)
- GPIO_Out_En
 - Типы, [118](#)
 - niietcm4_gpio.h, [486](#)
- GPIO_Out_TypeDef
 - Типы, [118](#)
 - niietcm4_gpio.h, [485](#)
- GPIO_OutMode
 - GPIO_Init_TypeDef, [345](#)
- GPIO_OutMode_OD
 - Типы, [118](#)
 - niietcm4_gpio.h, [486](#)
- GPIO_OutMode_PP
 - Типы, [118](#)
 - niietcm4_gpio.h, [486](#)
- GPIO_OutMode_TypeDef
 - Типы, [118](#)
 - niietcm4_gpio.h, [486](#)
- GPIO_Pin
 - GPIO_Init_TypeDef, [345](#)
- GPIO_Pin_0
 - Маски пинов, [121](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_0_3
 - Маски пинов, [121](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_0_7
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_1
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_10
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_11
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_12
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [479](#)
- GPIO_Pin_12_15
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_13
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_14
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_15
 - Маски пинов, [122](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_2
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_3
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_4
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_4_7
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_5
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [480](#)
- GPIO_Pin_6
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [481](#)
- GPIO_Pin_7
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [481](#)
- GPIO_Pin_8
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [481](#)
- GPIO_Pin_8_11
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [481](#)
- GPIO_Pin_8_15
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [481](#)
- GPIO_Pin_9
 - Маски пинов, [123](#)
 - niietcm4_gpio.h, [481](#)

- Маски пинов, [124](#)
- [niietcm4_gpio.h](#), [481](#)
- GPIO_Pin_All
 - Маски пинов, [124](#)
 - [niietcm4_gpio.h](#), [481](#)
- GPIO_PullUp
 - GPIO_Init_TypeDef, [345](#)
- GPIO_PullUp_Dis
 - Типы, [118](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_PullUp_En
 - Типы, [118](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_PullUp_TypeDef
 - Типы, [118](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_Qual_Dis
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_Qual_En
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_Qual_TypeDef
 - Типы, [118](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_QualCmd
 - Фильтрация, [135](#)
 - Приватные функции, [273](#)
 - [niietcm4_gpio.c](#), [471](#)
 - [niietcm4_gpio.h](#), [489](#)
- GPIO_QualConfig
 - Фильтрация, [135](#)
 - Приватные функции, [273](#)
 - [niietcm4_gpio.c](#), [471](#)
 - [niietcm4_gpio.h](#), [489](#)
- GPIO_QualMode_3sample
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_QualMode_6sample
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_QualMode_TypeDef
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_Read
 - Чтение, [129](#)
 - Приватные функции, [274](#)
 - [niietcm4_gpio.c](#), [472](#)
 - [niietcm4_gpio.h](#), [490](#)
- GPIO_ReadBit
 - Чтение, [129](#)
 - Приватные функции, [274](#)
 - [niietcm4_gpio.c](#), [472](#)
 - [niietcm4_gpio.h](#), [490](#)
- GPIO_ReadMask
 - Чтение, [130](#)
 - Приватные функции, [274](#)
 - [niietcm4_gpio.c](#), [472](#)
- [niietcm4_gpio.h](#), [490](#)
- GPIO_Regs_A_C_E_G_Mask
 - Маски портов, [268](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_B_D_F_H_Mask
 - Маски портов, [268](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_GPIOA_Mask
 - Маски портов, [268](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_GPIOB_Mask
 - Маски портов, [268](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_GPIOC_Mask
 - Маски портов, [269](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_GPIOD_Mask
 - Маски портов, [269](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_GPIOE_Mask
 - Маски портов, [269](#)
 - [niietcm4_gpio.c](#), [467](#)
- GPIO_Regs_GPIOF_Mask
 - Маски портов, [269](#)
 - [niietcm4_gpio.c](#), [468](#)
- GPIO_Regs_GPIOG_Mask
 - Маски портов, [269](#)
 - [niietcm4_gpio.c](#), [468](#)
- GPIO_Regs_GPIOH_Mask
 - Маски портов, [269](#)
 - [niietcm4_gpio.c](#), [468](#)
- GPIO_SetBits
 - Битовые операции, [133](#)
 - Приватные функции, [275](#)
 - [niietcm4_gpio.c](#), [473](#)
 - [niietcm4_gpio.h](#), [491](#)
- GPIO_StructInit
 - Инициализация и деинициализация, [127](#)
 - Приватные функции, [275](#)
 - [niietcm4_gpio.c](#), [473](#)
 - [niietcm4_gpio.h](#), [491](#)
- GPIO_Sync_Dis
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [487](#)
- GPIO_Sync_En
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [487](#)
- GPIO_Sync_TypeDef
 - Типы, [119](#)
 - [niietcm4_gpio.h](#), [486](#)
- GPIO_SyncCmd
 - Фильтрация, [136](#)
 - Приватные функции, [275](#)
 - [niietcm4_gpio.c](#), [473](#)
 - [niietcm4_gpio.h](#), [491](#)
- GPIO_ToggleBits
 - Битовые операции, [134](#)
 - Приватные функции, [276](#)

- niietcm4_gpio.c, [474](#)
 - niietcm4_gpio.h, [492](#)
- GPIO_Write
 - Приватные функции, [276](#)
 - Запись, [131](#)
 - niietcm4_gpio.c, [474](#)
 - niietcm4_gpio.h, [492](#)
- GPIO_WriteBit
 - Приватные функции, [276](#)
 - Запись, [131](#)
 - niietcm4_gpio.c, [474](#)
 - niietcm4_gpio.h, [492](#)
- GPIO_WriteMask
 - Приватные функции, [277](#)
 - Запись, [132](#)
 - niietcm4_gpio.c, [475](#)
 - niietcm4_gpio.h, [493](#)
- INT_OSC_VALUE
 - Настройка драйвера, [321](#)
 - niietcm4.h, [355](#)
- IS_ADC_AVERAGE
 - Типы, [28](#)
 - niietcm4_adc.h, [388](#)
- IS_ADC_DC_CHANNEL
 - Типы, [28](#)
 - niietcm4_adc.h, [388](#)
- IS_ADC_DC_CONDITION
 - Типы, [29](#)
 - niietcm4_adc.h, [388](#)
- IS_ADC_DC_MODE
 - Типы, [29](#)
 - niietcm4_adc.h, [389](#)
- IS_ADC_DC_MODULE
 - Типы, [29](#)
 - niietcm4_adc.h, [389](#)
- IS_ADC_MEASURE
 - Типы, [30](#)
 - niietcm4_adc.h, [389](#)
- IS_ADC_MODE
 - Типы, [30](#)
 - niietcm4_adc.h, [389](#)
- IS_ADC_MODULE
 - Типы, [30](#)
 - niietcm4_adc.h, [390](#)
- IS_ADC_RESOLUTION
 - Типы, [31](#)
 - niietcm4_adc.h, [390](#)
- IS_ADC_SEQ_FIFO_LEVEL
 - Типы, [31](#)
 - niietcm4_adc.h, [390](#)
- IS_ADC_SEQ_MODULE
 - Типы, [31](#)
 - niietcm4_adc.h, [390](#)
- IS_ADC_SEQ_START_EVENT
 - Типы, [32](#)
 - niietcm4_adc.h, [391](#)
- IS_BOOTFLASH_STATUS
 - Типы, [64](#)
- niietcm4_bootflash.h, [418](#)
- IS_DMA_ARBITRATION_RATE
 - Типы, [87](#)
 - niietcm4_dma.h, [443](#)
- IS_DMA_DATA_INC
 - Типы, [87](#)
 - niietcm4_dma.h, [444](#)
- IS_DMA_DATA_SIZE
 - Типы, [88](#)
 - niietcm4_dma.h, [444](#)
- IS_DMA_MODE
 - Типы, [88](#)
 - niietcm4_dma.h, [444](#)
- IS_DMA_STATE
 - Типы, [88](#)
 - niietcm4_dma.h, [444](#)
- IS_EXTMEM_READ_WAITSTATE
 - Типы, [107](#)
 - niietcm4_extmem.h, [459](#)
- IS_EXTMEM_RW_WAITSTATE
 - Типы, [107](#)
 - niietcm4_extmem.h, [459](#)
- IS_EXTMEM_WIDTH
 - Типы, [107](#)
 - niietcm4_extmem.h, [459](#)
- IS_EXTMEM_WRITE_WAITSTATE
 - Типы, [107](#)
 - niietcm4_extmem.h, [459](#)
- IS_GET_DMA_CHANNEL
 - Маски каналов DMA, [76](#)
 - niietcm4_dma.h, [445](#)
- IS_GET_GPIO_PIN
 - Маски пинов, [124](#)
 - niietcm4_gpio.h, [481](#)
- IS_GPIO_ALL_PERIPH
 - Типы, [323](#)
 - niietcm4.h, [355](#)
- IS_GPIO_ALT_FUNC
 - Типы, [114](#)
 - niietcm4_gpio.h, [482](#)
- IS_GPIO_DIR
 - Типы, [114](#)
 - niietcm4_gpio.h, [482](#)
- IS_GPIO_INT_POL
 - Типы, [114](#)
 - niietcm4_gpio.h, [482](#)
- IS_GPIO_INT_TYPE
 - Типы, [115](#)
 - niietcm4_gpio.h, [482](#)
- IS_GPIO_LOAD
 - Типы, [115](#)
 - niietcm4_gpio.h, [482](#)
- IS_GPIO_MODE
 - Типы, [115](#)
 - niietcm4_gpio.h, [483](#)
- IS_GPIO_OUT
 - Типы, [115](#)
 - niietcm4_gpio.h, [483](#)

- IS_GPIO_OUT_MODE
 - Типы, 115
 - niietcm4_gpio.h, 483
- IS_GPIO_PULLUP
 - Типы, 116
 - niietcm4_gpio.h, 483
- IS_GPIO_QUAL
 - Типы, 116
 - niietcm4_gpio.h, 483
- IS_GPIO_QUAL_MODE
 - Типы, 116
 - niietcm4_gpio.h, 484
- IS_GPIO_SYNC
 - Типы, 116
 - niietcm4_gpio.h, 484
- IS_RCC_ADC_CLK
 - Типы, 142
 - niietcm4_rcc.h, 508
- IS_RCC_PERIPH_CLK
 - Типы, 142
 - niietcm4_rcc.h, 509
- IS_RCC_PLL_NO
 - Типы, 143
 - niietcm4_rcc.h, 509
- IS_RCC_PLL_REF
 - Типы, 143
 - niietcm4_rcc.h, 509
- IS_RCC_SPI_CLK
 - Типы, 143
 - niietcm4_rcc.h, 510
- IS_RCC_SYS_CLK
 - Типы, 143
 - niietcm4_rcc.h, 510
- IS_RCC_UART_CLK
 - Типы, 143
 - niietcm4_rcc.h, 510
- IS_RCC_USB_CLK
 - Типы, 144
 - niietcm4_rcc.h, 510
- IS_RCC_USB_FREQ
 - Типы, 144
 - niietcm4_rcc.h, 510
- IS_RTC_FORMAT
 - Типы, 166
 - niietcm4_rtc.h, 526
- IS_RTC_MONTH
 - Типы, 166
 - niietcm4_rtc.h, 526
- IS_RTC_WEEKDAY
 - Типы, 166
 - niietcm4_rtc.h, 527
- IS_SPI_ALL_PERIPH
 - Типы, 324
 - niietcm4.h, 356
- IS_TIMER_ALL_PERIPH
 - Типы, 324
 - niietcm4.h, 356
- IS_TIMER_EXT_INPUT
 - Типы, 169
 - niietcm4_timer.h, 535
- IS_UART_ALL_PERIPH
 - Типы, 324
 - niietcm4.h, 356
- IS_UART_DATA_WIDTH
 - Типы, 180
 - niietcm4_uart.h, 552
- IS_UART_DIR
 - Типы, 180
 - niietcm4_uart.h, 552
- IS_UART_ERROR
 - Типы, 181
 - niietcm4_uart.h, 553
- IS_UART_FIFO_LEVEL
 - Типы, 181
 - niietcm4_uart.h, 553
- IS_UART_FLAG
 - Типы, 181
 - niietcm4_uart.h, 553
- IS_UART_GET_IT_SOURCE
 - Типы, 181
 - niietcm4_uart.h, 553
- IS_UART_PARITY_BIT
 - Типы, 182
 - niietcm4_uart.h, 554
- IS_UART_STOP_BIT
 - Типы, 182
 - niietcm4_uart.h, 554
- IS_USERFLASH_STATUS
 - Типы, 203
 - niietcm4_userflash.h, 570
- N_MINUS_1
 - _CHANNEL_CFG_bits, 328
- NEXT_USEBURST
 - _CHANNEL_CFG_bits, 328
- niietcm4.h, 353
 - EXT_OSC_VALUE, 355
 - INT_OSC_VALUE, 355
 - IS_GPIO_ALL_PERIPH, 355
 - IS_SPI_ALL_PERIPH, 356
 - IS_TIMER_ALL_PERIPH, 356
 - IS_UART_ALL_PERIPH, 356
- niietcm4_adc.c, 356
 - ADC_Cmd, 359
 - ADC_DC_Cmd, 360
 - ADC_DC_DeInit, 360
 - ADC_DC_GetLastData, 360
 - ADC_DC_ITCmd, 361
 - ADC_DC_ITConfig, 361
 - ADC_DC_ITGenCmd, 362
 - ADC_DC_ITMaskCmd, 362
 - ADC_DC_ITMaskedStatus, 362
 - ADC_DC_ITRawStatus, 363
 - ADC_DC_ITStatusClear, 363
 - ADC_DC_Init, 361
 - ADC_DC_StructInit, 363
 - ADC_DC_TrigStatus, 364

- ADC_DC_TrigStatusClear, [364](#)
- ADC_DeInit, [364](#)
- ADC_Init, [364](#)
- ADC_SEQ_Cmd, [366](#)
- ADC_SEQ_DMAMCmd, [366](#)
- ADC_SEQ_DMAConfig, [367](#)
- ADC_SEQ_DMAErrorStatus, [367](#)
- ADC_SEQ_DMAErrorStatusClear, [367](#)
- ADC_SEQ_DeInit, [366](#)
- ADC_SEQ_FIFOEmptyStatus, [368](#)
- ADC_SEQ_FIFOEmptyStatusClear, [368](#)
- ADC_SEQ_FIFOFullStatus, [368](#)
- ADC_SEQ_FIFOFullStatusClear, [369](#)
- ADC_SEQ_GetConversionCount, [369](#)
- ADC_SEQ_GetFIFOData, [369](#)
- ADC_SEQ_GetFIFOLoad, [369](#)
- ADC_SEQ_GetITCount, [371](#)
- ADC_SEQ_ITCmd, [371](#)
- ADC_SEQ_ITConfig, [372](#)
- ADC_SEQ_ITCountRst, [372](#)
- ADC_SEQ_ITMaskedStatus, [372](#)
- ADC_SEQ_ITRawStatus, [373](#)
- ADC_SEQ_ITStatusClear, [373](#)
- ADC_SEQ_Init, [371](#)
- ADC_SEQ_SWReq, [374](#)
- ADC_SEQ_StructInit, [373](#)
- ADC_StructInit, [374](#)
- niietcm4_adc.h, [374](#)
- ADC_Average_16, [392](#)
- ADC_Average_2, [391](#)
- ADC_Average_32, [392](#)
- ADC_Average_4, [392](#)
- ADC_Average_64, [392](#)
- ADC_Average_8, [392](#)
- ADC_Average_Disable, [391](#)
- ADC_Average_TypeDef, [391](#)
- ADC_Channel_0, [381](#)
- ADC_Channel_1, [381](#)
- ADC_Channel_10, [381](#)
- ADC_Channel_11, [381](#)
- ADC_Channel_12, [382](#)
- ADC_Channel_13, [382](#)
- ADC_Channel_14, [382](#)
- ADC_Channel_15, [382](#)
- ADC_Channel_16, [382](#)
- ADC_Channel_17, [382](#)
- ADC_Channel_18, [382](#)
- ADC_Channel_19, [382](#)
- ADC_Channel_2, [382](#)
- ADC_Channel_20, [383](#)
- ADC_Channel_21, [383](#)
- ADC_Channel_22, [383](#)
- ADC_Channel_23, [383](#)
- ADC_Channel_3, [383](#)
- ADC_Channel_4, [383](#)
- ADC_Channel_5, [383](#)
- ADC_Channel_6, [383](#)
- ADC_Channel_7, [383](#)
- ADC_Channel_8, [383](#)
- ADC_Channel_9, [384](#)
- ADC_Channel_All, [384](#)
- ADC_Channel_None, [384](#)
- ADC_Cmd, [396](#)
- ADC_DC_0, [384](#)
- ADC_DC_1, [384](#)
- ADC_DC_10, [384](#)
- ADC_DC_11, [384](#)
- ADC_DC_12, [384](#)
- ADC_DC_13, [384](#)
- ADC_DC_14, [385](#)
- ADC_DC_15, [385](#)
- ADC_DC_16, [385](#)
- ADC_DC_17, [385](#)
- ADC_DC_18, [385](#)
- ADC_DC_19, [385](#)
- ADC_DC_2, [385](#)
- ADC_DC_20, [385](#)
- ADC_DC_21, [385](#)
- ADC_DC_22, [386](#)
- ADC_DC_23, [386](#)
- ADC_DC_3, [386](#)
- ADC_DC_4, [386](#)
- ADC_DC_5, [386](#)
- ADC_DC_6, [386](#)
- ADC_DC_7, [386](#)
- ADC_DC_8, [386](#)
- ADC_DC_9, [386](#)
- ADC_DC_All, [386](#)
- ADC_DC_Channel_0, [392](#)
- ADC_DC_Channel_1, [392](#)
- ADC_DC_Channel_10, [392](#)
- ADC_DC_Channel_11, [392](#)
- ADC_DC_Channel_12, [392](#)
- ADC_DC_Channel_13, [392](#)
- ADC_DC_Channel_14, [392](#)
- ADC_DC_Channel_15, [392](#)
- ADC_DC_Channel_16, [392](#)
- ADC_DC_Channel_17, [392](#)
- ADC_DC_Channel_18, [392](#)
- ADC_DC_Channel_19, [392](#)
- ADC_DC_Channel_2, [392](#)
- ADC_DC_Channel_20, [392](#)
- ADC_DC_Channel_21, [392](#)
- ADC_DC_Channel_22, [392](#)
- ADC_DC_Channel_23, [392](#)
- ADC_DC_Channel_3, [392](#)
- ADC_DC_Channel_4, [392](#)
- ADC_DC_Channel_5, [392](#)
- ADC_DC_Channel_6, [392](#)
- ADC_DC_Channel_7, [392](#)
- ADC_DC_Channel_8, [392](#)
- ADC_DC_Channel_9, [392](#)
- ADC_DC_Channel_None, [392](#)
- ADC_DC_Channel_TypeDef, [392](#)
- ADC_DC_Cmd, [396](#)
- ADC_DC_Condition_High, [393](#)

- ADC_DC_Condition_Low, 393
- ADC_DC_Condition_TypeDef, 392
- ADC_DC_Condition_Window, 393
- ADC_DC_DeInit, 396
- ADC_DC_GetLastData, 397
- ADC_DC_ITCmd, 397
- ADC_DC_ITConfig, 398
- ADC_DC_ITGenCmd, 398
- ADC_DC_ITMaskCmd, 398
- ADC_DC_ITMaskedStatus, 400
- ADC_DC_ITRawStatus, 400
- ADC_DC_ITStatusClear, 400
- ADC_DC_Init, 397
- ADC_DC_Mode_Multiple, 393
- ADC_DC_Mode_MultipleHyst, 393
- ADC_DC_Mode_Single, 393
- ADC_DC_Mode_SingleHyst, 393
- ADC_DC_Mode_TypeDef, 393
- ADC_DC_Module_0, 393
- ADC_DC_Module_1, 393
- ADC_DC_Module_10, 393
- ADC_DC_Module_11, 393
- ADC_DC_Module_12, 393
- ADC_DC_Module_13, 393
- ADC_DC_Module_14, 393
- ADC_DC_Module_15, 393
- ADC_DC_Module_16, 393
- ADC_DC_Module_17, 393
- ADC_DC_Module_18, 393
- ADC_DC_Module_19, 394
- ADC_DC_Module_2, 393
- ADC_DC_Module_20, 394
- ADC_DC_Module_21, 394
- ADC_DC_Module_22, 394
- ADC_DC_Module_23, 394
- ADC_DC_Module_3, 393
- ADC_DC_Module_4, 393
- ADC_DC_Module_5, 393
- ADC_DC_Module_6, 393
- ADC_DC_Module_7, 393
- ADC_DC_Module_8, 393
- ADC_DC_Module_9, 393
- ADC_DC_Module_TypeDef, 393
- ADC_DC_None, 387
- ADC_DC_StructInit, 401
- ADC_DC_TrigStatus, 401
- ADC_DC_TrigStatusClear, 401
- ADC_DeInit, 402
- ADC_Init, 402
- ADC_Measure_Diff, 394
- ADC_Measure_Single, 394
- ADC_Measure_TypeDef, 394
- ADC_Mode_Active, 394
- ADC_Mode_Powerdown, 394
- ADC_Mode_StandBy, 394
- ADC_Mode_TypeDef, 394
- ADC_Module_0, 394
- ADC_Module_1, 394
- ADC_Module_10, 394
- ADC_Module_11, 394
- ADC_Module_2, 394
- ADC_Module_3, 394
- ADC_Module_4, 394
- ADC_Module_5, 394
- ADC_Module_6, 394
- ADC_Module_7, 394
- ADC_Module_8, 394
- ADC_Module_9, 394
- ADC_Module_TypeDef, 394
- ADC_Resolution_10bit, 395
- ADC_Resolution_12bit, 395
- ADC_Resolution_TypeDef, 394
- ADC_SEQ_0, 387
- ADC_SEQ_1, 387
- ADC_SEQ_2, 387
- ADC_SEQ_3, 387
- ADC_SEQ_4, 387
- ADC_SEQ_5, 387
- ADC_SEQ_6, 387
- ADC_SEQ_7, 387
- ADC_SEQ_Cmd, 402
- ADC_SEQ_DMAMCmd, 403
- ADC_SEQ_DMAConfig, 403
- ADC_SEQ_DMAErrorStatus, 404
- ADC_SEQ_DMAErrorStatusClear, 404
- ADC_SEQ_DeInit, 403
- ADC_SEQ_FIFOEmptyStatus, 404
- ADC_SEQ_FIFOEmptyStatusClear, 404
- ADC_SEQ_FIFOFullStatus, 406
- ADC_SEQ_FIFOFullStatusClear, 406
- ADC_SEQ_FIFOLevel_1, 395
- ADC_SEQ_FIFOLevel_16, 395
- ADC_SEQ_FIFOLevel_2, 395
- ADC_SEQ_FIFOLevel_32, 395
- ADC_SEQ_FIFOLevel_4, 395
- ADC_SEQ_FIFOLevel_8, 395
- ADC_SEQ_FIFOLevel_TypeDef, 395
- ADC_SEQ_GetConversionCount, 406
- ADC_SEQ_GetFIFOData, 407
- ADC_SEQ_GetFIFOLoad, 407
- ADC_SEQ_GetITCount, 407
- ADC_SEQ_ITCmd, 409
- ADC_SEQ_ITConfig, 409
- ADC_SEQ_ITCountRst, 409
- ADC_SEQ_ITMaskedStatus, 410
- ADC_SEQ_ITRawStatus, 410
- ADC_SEQ_ITStatusClear, 410
- ADC_SEQ_Init, 407
- ADC_SEQ_Module_0, 395
- ADC_SEQ_Module_1, 395
- ADC_SEQ_Module_2, 395
- ADC_SEQ_Module_3, 395
- ADC_SEQ_Module_4, 395
- ADC_SEQ_Module_5, 395
- ADC_SEQ_Module_6, 395
- ADC_SEQ_Module_7, 395

- ADC_SEQ_Module_TypeDef, 395
- ADC_SEQ_SWReq, 411
- ADC_SEQ_StartEvent_CMP0, 395
- ADC_SEQ_StartEvent_CMP1, 395
- ADC_SEQ_StartEvent_CMP2, 396
- ADC_SEQ_StartEvent_Cycle, 396
- ADC_SEQ_StartEvent_ITGPIO, 396
- ADC_SEQ_StartEvent_PWM0, 396
- ADC_SEQ_StartEvent_PWM1, 396
- ADC_SEQ_StartEvent_PWM2, 396
- ADC_SEQ_StartEvent_PWM3, 396
- ADC_SEQ_StartEvent_PWM4, 396
- ADC_SEQ_StartEvent_PWM5, 396
- ADC_SEQ_StartEvent_SWReq, 395
- ADC_SEQ_StartEvent_TIM, 396
- ADC_SEQ_StartEvent_TypeDef, 395
- ADC_SEQ_StructInit, 411
- ADC_StructInit, 411
- IS_ADC_AVERAGE, 388
- IS_ADC_DC_CHANNEL, 388
- IS_ADC_DC_CONDITION, 388
- IS_ADC_DC_MODE, 389
- IS_ADC_DC_MODULE, 389
- IS_ADC_MEASURE, 389
- IS_ADC_MODE, 389
- IS_ADC_MODULE, 390
- IS_ADC_RESOLUTION, 390
- IS_ADC_SEQ_FIFO_LEVEL, 390
- IS_ADC_SEQ_MODULE, 390
- IS_ADC_SEQ_START_EVENT, 391
- niietcm4_bootflash.c, 411
- BOOTFLASH_FullErase, 413
- BOOTFLASH_ITCmd, 414
- BOOTFLASH_Info_PageErase, 413
- BOOTFLASH_Info_Write, 413
- BOOTFLASH_Init, 414
- BOOTFLASH_OperationStatus, 414
- BOOTFLASH_OperationStatusClear, 414
- BOOTFLASH_PageErase, 415
- BOOTFLASH_Write, 415
- niietcm4_bootflash.h, 415
- BOOTFLASH_FullErase, 419
- BOOTFLASH_INFO_PAGE_SIZE_BYT↔ES, 417
- BOOTFLASH_INFO_PAGE_TOTAL, 417
- BOOTFLASH_INFO_TOTAL_BYTES, 418
- BOOTFLASH_ITCmd, 420
- BOOTFLASH_Info_PageErase, 419
- BOOTFLASH_Info_Write, 419
- BOOTFLASH_Init, 419
- BOOTFLASH_OperationStatus, 420
- BOOTFLASH_OperationStatusClear, 420
- BOOTFLASH_PAGE_SIZE_BYTES, 418
- BOOTFLASH_PAGE_TOTAL, 418
- BOOTFLASH_PageErase, 420
- BOOTFLASH_Status_Complete, 418
- BOOTFLASH_Status_Error, 418
- BOOTFLASH_Status_None, 418
- BOOTFLASH_Status_TypeDef, 418
- BOOTFLASH_TOTAL_BYTES, 418
- BOOTFLASH_Write, 420
- IS_BOOTFLASH_STATUS, 418
- niietcm4_conf.h, 422
- niietcm4_dma.c, 423
- DMA_BasePtrConfig, 425
- DMA_ChannelDeInit, 425
- DMA_ChannelEnableCmd, 425
- DMA_ChannelInit, 426
- DMA_ChannelStructInit, 426
- DMA_ClearErrorStatus, 427
- DMA_DeInit, 427
- DMA_ErrorStatus, 427
- DMA_HighPriorityCmd, 427
- DMA_Init, 428
- DMA_MasterEnableCmd, 428
- DMA_MasterEnableStatus, 428
- DMA_PrmAltCmd, 429
- DMA_ProtectionConfig, 429
- DMA_ReqMaskCmd, 429
- DMA_SWRequestCmd, 430
- DMA_StateStatus, 430
- DMA_StructInit, 430
- DMA_UseBurstCmd, 430
- DMA_WaitOnReqStatus, 431
- niietcm4_dma.h, 431
- CHANNEL_CFG_CYCLE_CTRL_Msk, 436
- CHANNEL_CFG_CYCLE_CTRL_Pos, 436
- CHANNEL_CFG_DST_INC_Msk, 436
- CHANNEL_CFG_DST_INC_Pos, 436
- CHANNEL_CFG_DST_PROT_CTRL_Msk, 436
- CHANNEL_CFG_DST_PROT_CTRL_Pos, 436
- CHANNEL_CFG_DST_SIZE_Msk, 436
- CHANNEL_CFG_DST_SIZE_Pos, 437
- CHANNEL_CFG_N_MINUS_1_Msk, 437
- CHANNEL_CFG_N_MINUS_1_Pos, 437
- CHANNEL_CFG_NEXT_USEBURST_Msk, 437
- CHANNEL_CFG_NEXT_USEBURST_Pos, 437
- CHANNEL_CFG_R_POWER_Msk, 437
- CHANNEL_CFG_R_POWER_Pos, 437
- CHANNEL_CFG_SRC_INC_Msk, 437
- CHANNEL_CFG_SRC_INC_Pos, 437
- CHANNEL_CFG_SRC_PROT_CTRL_Msk, 438
- CHANNEL_CFG_SRC_PROT_CTRL_Pos, 438
- CHANNEL_CFG_SRC_SIZE_Msk, 438
- CHANNEL_CFG_SRC_SIZE_Pos, 438
- DMA_ArbitrationRate_1, 445
- DMA_ArbitrationRate_1024, 446
- DMA_ArbitrationRate_128, 446
- DMA_ArbitrationRate_16, 446

- DMA_ArbitrationRate_2, [445](#)
- DMA_ArbitrationRate_256, [446](#)
- DMA_ArbitrationRate_32, [446](#)
- DMA_ArbitrationRate_4, [445](#)
- DMA_ArbitrationRate_512, [446](#)
- DMA_ArbitrationRate_64, [446](#)
- DMA_ArbitrationRate_8, [445](#)
- DMA_ArbitrationRate_TypeDef, [445](#)
- DMA_BasePtrConfig, [447](#)
- DMA_Channel_0, [438](#)
- DMA_Channel_1, [438](#)
- DMA_Channel_10, [438](#)
- DMA_Channel_11, [438](#)
- DMA_Channel_12, [438](#)
- DMA_Channel_13, [439](#)
- DMA_Channel_14, [439](#)
- DMA_Channel_15, [439](#)
- DMA_Channel_16, [439](#)
- DMA_Channel_17, [439](#)
- DMA_Channel_18, [439](#)
- DMA_Channel_19, [439](#)
- DMA_Channel_2, [439](#)
- DMA_Channel_20, [439](#)
- DMA_Channel_21, [439](#)
- DMA_Channel_22, [440](#)
- DMA_Channel_23, [440](#)
- DMA_Channel_3, [440](#)
- DMA_Channel_4, [440](#)
- DMA_Channel_5, [440](#)
- DMA_Channel_6, [440](#)
- DMA_Channel_7, [440](#)
- DMA_Channel_8, [440](#)
- DMA_Channel_9, [440](#)
- DMA_Channel_ADCSEQ0, [441](#)
- DMA_Channel_ADCSEQ1, [441](#)
- DMA_Channel_ADCSEQ2, [441](#)
- DMA_Channel_ADCSEQ3, [441](#)
- DMA_Channel_ADCSEQ4, [441](#)
- DMA_Channel_ADCSEQ5, [441](#)
- DMA_Channel_ADCSEQ6, [441](#)
- DMA_Channel_ADCSEQ7, [441](#)
- DMA_Channel_All, [441](#)
- DMA_Channel_SPI0_RX, [441](#)
- DMA_Channel_SPI0_TX, [442](#)
- DMA_Channel_SPI1_RX, [442](#)
- DMA_Channel_SPI1_TX, [442](#)
- DMA_Channel_SPI2_RX, [442](#)
- DMA_Channel_SPI2_TX, [442](#)
- DMA_Channel_SPI3_RX, [442](#)
- DMA_Channel_SPI3_TX, [442](#)
- DMA_Channel_UART0_RX, [442](#)
- DMA_Channel_UART0_TX, [442](#)
- DMA_Channel_UART1_RX, [443](#)
- DMA_Channel_UART1_TX, [443](#)
- DMA_Channel_UART2_RX, [443](#)
- DMA_Channel_UART2_TX, [443](#)
- DMA_Channel_UART3_RX, [443](#)
- DMA_Channel_UART3_TX, [443](#)
- DMA_ChannelDeInit, [447](#)
- DMA_ChannelEnableCmd, [447](#)
- DMA_ChannelInit, [448](#)
- DMA_ChannelStructInit, [448](#)
- DMA_ClearErrorStatus, [449](#)
- DMA_DataInc_16, [446](#)
- DMA_DataInc_32, [446](#)
- DMA_DataInc_8, [446](#)
- DMA_DataInc_Disable, [446](#)
- DMA_DataInc_TypeDef, [446](#)
- DMA_DataSize_16, [446](#)
- DMA_DataSize_32, [446](#)
- DMA_DataSize_8, [446](#)
- DMA_DataSize_TypeDef, [446](#)
- DMA_DeInit, [449](#)
- DMA_ErrorStatus, [449](#)
- DMA_HighPriorityCmd, [449](#)
- DMA_Init, [450](#)
- DMA_MasterEnableCmd, [450](#)
- DMA_MasterEnableStatus, [450](#)
- DMA_Mode_AltMemScatGath, [446](#)
- DMA_Mode_AltPeriphScatGath, [446](#)
- DMA_Mode_AutoReq, [446](#)
- DMA_Mode_Basic, [446](#)
- DMA_Mode_Disable, [446](#)
- DMA_Mode_PingPong, [446](#)
- DMA_Mode_PrmMemScatGath, [446](#)
- DMA_Mode_PrmPeriphScatGath, [446](#)
- DMA_Mode_TypeDef, [446](#)
- DMA_PrmAltCmd, [451](#)
- DMA_ProtectionConfig, [451](#)
- DMA_ReqMaskCmd, [451](#)
- DMA_SWRequestCmd, [452](#)
- DMA_State_Done, [447](#)
- DMA_State_Free, [447](#)
- DMA_State_Pause, [447](#)
- DMA_State_PeriphScatGath, [447](#)
- DMA_State_ReadConfigData, [447](#)
- DMA_State_ReadDstDataEndPtr, [447](#)
- DMA_State_ReadSrcData, [447](#)
- DMA_State_ReadSrcDataEndPtr, [447](#)
- DMA_State_TypeDef, [446](#)
- DMA_State_WaitReq, [447](#)
- DMA_State_WriteDstData, [447](#)
- DMA_StateStatus, [452](#)
- DMA_StructInit, [452](#)
- DMA_UseBurstCmd, [452](#)
- DMA_WaitOnReqStatus, [453](#)
- IS_DMA_ARBITRATION_RATE, [443](#)
- IS_DMA_DATA_INC, [444](#)
- IS_DMA_DATA_SIZE, [444](#)
- IS_DMA_MODE, [444](#)
- IS_DMA_STATE, [444](#)
- IS_GET_DMA_CHANNEL, [445](#)
- niietcm4_extmem.c, [453](#)
- EXT_MEM_CFG_Reset_Value, [454](#)
- EXTMEM_DeInit, [454](#)
- EXTMEM_Init, [455](#)

- EXTMEM_StructInit, 455
- niietcm4_extmem.h, 455
 - EXTMEM_CEMask_Addr_11, 458
 - EXTMEM_CEMask_Addr_11_19, 458
 - EXTMEM_CEMask_Addr_12, 458
 - EXTMEM_CEMask_Addr_13, 458
 - EXTMEM_CEMask_Addr_14, 458
 - EXTMEM_CEMask_Addr_15, 458
 - EXTMEM_CEMask_Addr_16, 458
 - EXTMEM_CEMask_Addr_17, 458
 - EXTMEM_CEMask_Addr_18, 458
 - EXTMEM_CEMask_Addr_19, 459
 - EXTMEM_DeInit, 461
 - EXTMEM_Init, 461
 - EXTMEM_RWWaitState_1, 460
 - EXTMEM_RWWaitState_2, 460
 - EXTMEM_RWWaitState_3, 460
 - EXTMEM_RWWaitState_4, 460
 - EXTMEM_RWWaitState_5, 460
 - EXTMEM_RWWaitState_6, 460
 - EXTMEM_RWWaitState_7, 460
 - EXTMEM_RWWaitState_8, 460
 - EXTMEM_RWWaitState_TypeDef, 460
 - EXTMEM_ReadWaitState_1, 460
 - EXTMEM_ReadWaitState_2, 460
 - EXTMEM_ReadWaitState_3, 460
 - EXTMEM_ReadWaitState_4, 460
 - EXTMEM_ReadWaitState_5, 460
 - EXTMEM_ReadWaitState_6, 460
 - EXTMEM_ReadWaitState_7, 460
 - EXTMEM_ReadWaitState_8, 460
 - EXTMEM_ReadWaitState_TypeDef, 460
 - EXTMEM_StructInit, 461
 - EXTMEM_Width_16bit, 461
 - EXTMEM_Width_8bit, 461
 - EXTMEM_Width_TypeDef, 460
 - EXTMEM_WriteWaitState_1, 461
 - EXTMEM_WriteWaitState_2, 461
 - EXTMEM_WriteWaitState_3, 461
 - EXTMEM_WriteWaitState_4, 461
 - EXTMEM_WriteWaitState_5, 461
 - EXTMEM_WriteWaitState_6, 461
 - EXTMEM_WriteWaitState_7, 461
 - EXTMEM_WriteWaitState_8, 461
 - EXTMEM_WriteWaitState_TypeDef, 461
- IS_EXTMEM_READ_WAITSTATE, 459
- IS_EXTMEM_RW_WAITSTATE, 459
- IS_EXTMEM_WIDTH, 459
- IS_EXTMEM_WRITE_WAITSTATE, 459
- niietcm4_gpio.c, 463
 - GPIO_ClearBits, 468
 - GPIO_DATAOUT_Reset_Value, 465
 - GPIO_DeInit, 468
 - GPIO_GPIODEN0_Reset_Value, 465
 - GPIO_GPIODEN1_Reset_Value, 465
 - GPIO_GPIODEN2_Reset_Value, 465
 - GPIO_GPIODEN3_Reset_Value, 466
 - GPIO_GPIOODCTLx_Reset_Value, 466
 - GPIO_GPIOODSCTLx_Reset_Value, 466
 - GPIO_GPIOPCTLx_Reset_Value, 466
 - GPIO_GPIOPUCTLx_Reset_Value, 466
 - GPIO_GPIOQEx_Reset_Value, 466
 - GPIO_GPIOQMx_Reset_Value, 466
 - GPIO_GPIOQPX_Reset_Value, 466
 - GPIO_GPIOSEx_Reset_Value, 467
 - GPIO_ITCmd, 470
 - GPIO_ITConfig, 470
 - GPIO_ITStatusClear, 471
 - GPIO_Init, 468
 - GPIO_QualCmd, 471
 - GPIO_QualConfig, 471
 - GPIO_Read, 472
 - GPIO_ReadBit, 472
 - GPIO_ReadMask, 472
 - GPIO_Regs_A_C_E_G_Mask, 467
 - GPIO_Regs_B_D_F_H_Mask, 467
 - GPIO_Regs_GPIOA_Mask, 467
 - GPIO_Regs_GPIOB_Mask, 467
 - GPIO_Regs_GPIOC_Mask, 467
 - GPIO_Regs_GPIOD_Mask, 467
 - GPIO_Regs_GPIOE_Mask, 467
 - GPIO_Regs_GPIOF_Mask, 468
 - GPIO_Regs_GPIOG_Mask, 468
 - GPIO_Regs_GPIOH_Mask, 468
 - GPIO_SetBits, 473
 - GPIO_StructInit, 473
 - GPIO_SyncCmd, 473
 - GPIO_ToggleBits, 474
 - GPIO_Write, 474
 - GPIO_WriteBit, 474
 - GPIO_WriteMask, 475
- niietcm4_gpio.h, 475
 - Bit_CLEAR, 484
 - Bit_SET, 484
 - BitAction, 484
 - GPIO_AltFunc_1, 484
 - GPIO_AltFunc_2, 484
 - GPIO_AltFunc_3, 484
 - GPIO_AltFunc_TypeDef, 484
 - GPIO_ClearBits, 487
 - GPIO_DeInit, 487
 - GPIO_Dir_In, 485
 - GPIO_Dir_Out, 485
 - GPIO_Dir_TypeDef, 484
 - GPIO_ITCmd, 488
 - GPIO_ITConfig, 488
 - GPIO_ITStatusClear, 489
 - GPIO_Init, 487
 - GPIO_IntPol_Neg, 485
 - GPIO_IntPol_Pos, 485
 - GPIO_IntPol_TypeDef, 485
 - GPIO_IntType_Edge, 485
 - GPIO_IntType_Level, 485
 - GPIO_IntType_TypeDef, 485
 - GPIO_Load_16mA, 485
 - GPIO_Load_8mA, 485

- GPIO_Load_TypeDef, 485
- GPIO_Mode_AltFunc, 485
- GPIO_Mode_IO, 485
- GPIO_Mode_TypeDef, 485
- GPIO_Out_Dis, 486
- GPIO_Out_En, 486
- GPIO_Out_TypeDef, 485
- GPIO_OutMode_OD, 486
- GPIO_OutMode_PP, 486
- GPIO_OutMode_TypeDef, 486
- GPIO_Pin_0, 479
- GPIO_Pin_0_3, 479
- GPIO_Pin_0_7, 479
- GPIO_Pin_1, 479
- GPIO_Pin_10, 479
- GPIO_Pin_11, 479
- GPIO_Pin_12, 479
- GPIO_Pin_12_15, 480
- GPIO_Pin_13, 480
- GPIO_Pin_14, 480
- GPIO_Pin_15, 480
- GPIO_Pin_2, 480
- GPIO_Pin_3, 480
- GPIO_Pin_4, 480
- GPIO_Pin_4_7, 480
- GPIO_Pin_5, 480
- GPIO_Pin_6, 481
- GPIO_Pin_7, 481
- GPIO_Pin_8, 481
- GPIO_Pin_8_11, 481
- GPIO_Pin_8_15, 481
- GPIO_Pin_9, 481
- GPIO_Pin_All, 481
- GPIO_PullUp_Dis, 486
- GPIO_PullUp_En, 486
- GPIO_PullUp_TypeDef, 486
- GPIO_Qual_Dis, 486
- GPIO_Qual_En, 486
- GPIO_Qual_TypeDef, 486
- GPIO_QualCmd, 489
- GPIO_QualConfig, 489
- GPIO_QualMode_3sample, 486
- GPIO_QualMode_6sample, 486
- GPIO_QualMode_TypeDef, 486
- GPIO_Read, 490
- GPIO_ReadBit, 490
- GPIO_ReadMask, 490
- GPIO_SetBits, 491
- GPIO_StructInit, 491
- GPIO_Sync_Dis, 487
- GPIO_Sync_En, 487
- GPIO_Sync_TypeDef, 486
- GPIO_SyncCmd, 491
- GPIO_ToggleBits, 492
- GPIO_Write, 492
- GPIO_WriteBit, 492
- GPIO_WriteMask, 493
- IS_GET_GPIO_PIN, 481
- IS_GPIO_ALT_FUNC, 482
- IS_GPIO_DIR, 482
- IS_GPIO_INT_POL, 482
- IS_GPIO_INT_TYPE, 482
- IS_GPIO_LOAD, 482
- IS_GPIO_MODE, 483
- IS_GPIO_OUT, 483
- IS_GPIO_OUT_MODE, 483
- IS_GPIO_PULLUP, 483
- IS_GPIO_QUAL, 483
- IS_GPIO_QUAL_MODE, 484
- IS_GPIO_SYNC, 484
- niietcm4_rcc.c, 493
 - RCC_ADCClkCmd, 496
 - RCC_ADCClkDivConfig, 496
 - RCC_PLL_CTRL_Reset_Value, 495
 - RCC_PLL_NF_Reset_Value, 495
 - RCC_PLL_NR_Reset_Value, 495
 - RCC_PLL_OD_Reset_Value, 495
 - RCC_PLLAutoConfig, 497
 - RCC_PLLDeInit, 498
 - RCC_PLLInit, 498
 - RCC_PLLPowerDownCmd, 499
 - RCC_PLLStructInit, 499
 - RCC_PeriphClkCmd, 496
 - RCC_PeriphRstCmd, 497
 - RCC_SPIClkCmd, 499
 - RCC_SPIClkDivConfig, 501
 - RCC_SPIClkSel, 501
 - RCC_SysClkDiv2Out, 501
 - RCC_SysClkSel, 502
 - RCC_SysClkStatus, 502
 - RCC_UARTClkCmd, 502
 - RCC_UARTClkDivConfig, 503
 - RCC_UARTClkSel, 503
 - RCC_USBClkCmd, 503
 - RCC_USBClkConfig, 504
 - RCC_WaitClkChange, 504
- niietcm4_rcc.h, 504
 - IS_RCC_ADC_CLK, 508
 - IS_RCC_PERIPH_CLK, 509
 - IS_RCC_PLL_NO, 509
 - IS_RCC_PLL_REF, 509
 - IS_RCC_SPI_CLK, 510
 - IS_RCC_SYS_CLK, 510
 - IS_RCC_UART_CLK, 510
 - IS_RCC_USB_CLK, 510
 - IS_RCC_USB_FREQ, 510
 - RCC_ADCClk_0, 511
 - RCC_ADCClk_1, 511
 - RCC_ADCClk_10, 511
 - RCC_ADCClk_11, 511
 - RCC_ADCClk_2, 511
 - RCC_ADCClk_3, 511
 - RCC_ADCClk_4, 511
 - RCC_ADCClk_5, 511
 - RCC_ADCClk_6, 511
 - RCC_ADCClk_7, 511

- RCC_ADCClk_8, [511](#)
- RCC_ADCClk_9, [511](#)
- RCC_ADCClk_TypeDef, [511](#)
- RCC_ADCClkCmd, [515](#)
- RCC_ADCClkDivConfig, [516](#)
- RCC_CLK_CHANGE_TIMEOUT, [511](#)
- RCC_PLLAutoConfig, [517](#)
- RCC_PLLDeInit, [518](#)
- RCC_PLLInit, [518](#)
- RCC_PLLNO_Disable, [513](#)
- RCC_PLLNO_Div2, [513](#)
- RCC_PLLNO_Div4, [513](#)
- RCC_PLLNO_TypeDef, [513](#)
- RCC_PLLPowerDownCmd, [519](#)
- RCC_PLLRef_ETH_25MHz, [513](#)
- RCC_PLLRef_TypeDef, [513](#)
- RCC_PLLRef_USB_60MHz, [513](#)
- RCC_PLLRef_USB_CLK, [513](#)
- RCC_PLLRef_XI_OSC, [513](#)
- RCC_PLLStructInit, [519](#)
- RCC_PeriphClk_ADC, [512](#)
- RCC_PeriphClk_CMP, [511](#)
- RCC_PeriphClk_I2C0, [512](#)
- RCC_PeriphClk_I2C1, [512](#)
- RCC_PeriphClk_PWM0, [511](#)
- RCC_PeriphClk_PWM1, [511](#)
- RCC_PeriphClk_PWM2, [512](#)
- RCC_PeriphClk_PWM3, [512](#)
- RCC_PeriphClk_PWM4, [512](#)
- RCC_PeriphClk_PWM5, [512](#)
- RCC_PeriphClk_PWM6, [512](#)
- RCC_PeriphClk_PWM7, [512](#)
- RCC_PeriphClk_PWM8, [512](#)
- RCC_PeriphClk_QEP0, [511](#)
- RCC_PeriphClk_QEP1, [511](#)
- RCC_PeriphClk_TypeDef, [511](#)
- RCC_PeriphClk_WD, [512](#)
- RCC_PeriphClkCmd, [516](#)
- RCC_PeriphRst_CAP0, [513](#)
- RCC_PeriphRst_CAP1, [513](#)
- RCC_PeriphRst_CAP2, [513](#)
- RCC_PeriphRst_CAP3, [513](#)
- RCC_PeriphRst_CAP4, [513](#)
- RCC_PeriphRst_CAP5, [513](#)
- RCC_PeriphRst_CMP, [513](#)
- RCC_PeriphRst_ETH, [512](#)
- RCC_PeriphRst_I2C0, [512](#)
- RCC_PeriphRst_I2C1, [512](#)
- RCC_PeriphRst_PWM0, [512](#)
- RCC_PeriphRst_PWM1, [512](#)
- RCC_PeriphRst_PWM2, [512](#)
- RCC_PeriphRst_PWM3, [512](#)
- RCC_PeriphRst_PWM4, [512](#)
- RCC_PeriphRst_PWM5, [512](#)
- RCC_PeriphRst_PWM6, [512](#)
- RCC_PeriphRst_PWM7, [512](#)
- RCC_PeriphRst_PWM8, [512](#)
- RCC_PeriphRst_QEP0, [512](#)
- RCC_PeriphRst_QEP1, [512](#)
- RCC_PeriphRst_SPI0, [512](#)
- RCC_PeriphRst_SPI1, [512](#)
- RCC_PeriphRst_SPI2, [512](#)
- RCC_PeriphRst_SPI3, [512](#)
- RCC_PeriphRst_Timer0, [512](#)
- RCC_PeriphRst_Timer1, [512](#)
- RCC_PeriphRst_Timer2, [512](#)
- RCC_PeriphRst_TypeDef, [512](#)
- RCC_PeriphRst_UART0, [512](#)
- RCC_PeriphRst_UART1, [512](#)
- RCC_PeriphRst_UART2, [512](#)
- RCC_PeriphRst_UART3, [512](#)
- RCC_PeriphRst_USB, [512](#)
- RCC_PeriphRst_WD, [512](#)
- RCC_PeriphRstCmd, [517](#)
- RCC_SPIClk_SYSCLK, [513](#)
- RCC_SPIClk_TypeDef, [513](#)
- RCC_SPIClk_USB_60MHz, [513](#)
- RCC_SPIClk_USB_CLK, [513](#)
- RCC_SPIClk_XI_OSC, [513](#)
- RCC_SPIClkCmd, [519](#)
- RCC_SPIClkDivConfig, [520](#)
- RCC_SPIClkSel, [520](#)
- RCC_SysClk_CPE_Sel, [514](#)
- RCC_SysClk_ETH25MHz, [514](#)
- RCC_SysClk_PLL, [514](#)
- RCC_SysClk_PLLDIV, [514](#)
- RCC_SysClk_POR, [514](#)
- RCC_SysClk_TypeDef, [513](#)
- RCC_SysClk_USB60MHz, [514](#)
- RCC_SysClk_USB_CLK, [514](#)
- RCC_SysClk_XI_OSC, [514](#)
- RCC_SysClkDiv2Out, [520](#)
- RCC_SysClkSel, [521](#)
- RCC_SysClkStatus, [521](#)
- RCC_UARTClk_SYSCLK, [514](#)
- RCC_UARTClk_TypeDef, [514](#)
- RCC_UARTClk_USB_60MHz, [514](#)
- RCC_UARTClk_USB_CLK, [514](#)
- RCC_UARTClk_XI_OSC, [514](#)
- RCC_UARTClkCmd, [521](#)
- RCC_UARTClkDivConfig, [521](#)
- RCC_UARTClkSel, [522](#)
- RCC_USBClk_TypeDef, [514](#)
- RCC_USBClk_USB_CLK, [514](#)
- RCC_USBClk_XI_OSC, [514](#)
- RCC_USBClkCmd, [522](#)
- RCC_USBClkConfig, [522](#)
- RCC_USBFreq_12MHz, [514](#)
- RCC_USBFreq_24MHz, [514](#)
- RCC_USBFreq_TypeDef, [514](#)
- niietcm4_rtc.c, [523](#)
- niietcm4_rtc.h, [524](#)
- IS_RTC_FORMAT, [526](#)
- IS_RTC_MONTH, [526](#)
- IS_RTC_WEEKDAY, [527](#)
- RTC_Format_BCD, [527](#)

- RTC_Format_BIN, 527
- RTC_Format_TypeDef, 527
- RTC_Month_April, 527
- RTC_Month_August, 527
- RTC_Month_December, 527
- RTC_Month_February, 527
- RTC_Month_January, 527
- RTC_Month_July, 527
- RTC_Month_June, 527
- RTC_Month_March, 527
- RTC_Month_May, 527
- RTC_Month_November, 527
- RTC_Month_October, 527
- RTC_Month_September, 527
- RTC_Month_TypeDef, 527
- RTC_Weekday_Friday, 528
- RTC_Weekday_Monday, 528
- RTC_Weekday_Saturday, 528
- RTC_Weekday_Sunday, 528
- RTC_Weekday_Thursday, 528
- RTC_Weekday_Tuesday, 528
- RTC_Weekday_TypeDef, 527
- RTC_Weekday_Wednesday, 528
- niietcm4_timer.c, 528
 - TIMER_Cmd, 529
 - TIMER_ExtInputConfig, 530
 - TIMER_FreqConfig, 530
 - TIMER_GetCounter, 530
 - TIMER_GetReload, 531
 - TIMER_ITCmd, 531
 - TIMER_ITStatus, 531
 - TIMER_ITStatusClear, 531
 - TIMER_PeriodConfig, 532
 - TIMER_SetCounter, 532
 - TIMER_SetReload, 532
- niietcm4_timer.h, 533
 - IS_TIMER_EXT_INPUT, 535
 - TIMER_Cmd, 535
 - TIMER_ExtInput_CountClk, 535
 - TIMER_ExtInput_CountEn, 535
 - TIMER_ExtInput_Disable, 535
 - TIMER_ExtInput_TypeDef, 535
 - TIMER_ExtInputConfig, 535
 - TIMER_FreqConfig, 536
 - TIMER_GetCounter, 536
 - TIMER_GetReload, 536
 - TIMER_ITCmd, 537
 - TIMER_ITStatus, 537
 - TIMER_ITStatusClear, 537
 - TIMER_PeriodConfig, 537
 - TIMER_SetCounter, 538
 - TIMER_SetReload, 538
- niietcm4_uart.c, 538
 - UART_BaudRateDivConfig, 541
 - UART_Break, 542
 - UART_Cmd, 542
 - UART_DMABlkOnErrCmd, 543
 - UART_DMACmd, 543
 - UART_DeInit, 542
 - UART_ErrorStatus, 543
 - UART_ErrorStatusClear, 543
 - UART_FlagStatus, 545
 - UART_ITCmd, 545
 - UART_ITFIFOLevelConfig, 546
 - UART_ITMaskedStatus, 546
 - UART_ITRawStatus, 546
 - UART_ITStatusClear, 547
 - UART_Init, 545
 - UART_ModemConfig, 547
 - UART_ModemStructInit, 547
 - UART_RecieveData, 548
 - UART_SendData, 548
 - UART_StructInit, 548
- niietcm4_uart.h, 549
 - IS_UART_DATA_WIDTH, 552
 - IS_UART_DIR, 552
 - IS_UART_ERROR, 553
 - IS_UART_FIFO_LEVEL, 553
 - IS_UART_FLAG, 553
 - IS_UART_GET_IT_SOURCE, 553
 - IS_UART_PARITY_BIT, 554
 - IS_UART_STOP_BIT, 554
 - UART_BaudRateDivConfig, 557
 - UART_Break, 557
 - UART_Cmd, 557
 - UART_DMABlkOnErrCmd, 558
 - UART_DMACmd, 558
 - UART_DataWidth_5, 555
 - UART_DataWidth_6, 555
 - UART_DataWidth_7, 555
 - UART_DataWidth_8, 555
 - UART_DataWidth_TypeDef, 554
 - UART_DeInit, 558
 - UART_Dir_Rx, 555
 - UART_Dir_Tx, 555
 - UART_Dir_Typedef, 555
 - UART_Error_Break, 555
 - UART_Error_Frame, 555
 - UART_Error_Overflow, 555
 - UART_Error_Parity, 555
 - UART_Error_Typedef, 555
 - UART_ErrorStatus, 558
 - UART_ErrorStatusClear, 559
 - UART_FIFOLevel_1_2, 555
 - UART_FIFOLevel_1_4, 555
 - UART_FIFOLevel_1_8, 555
 - UART_FIFOLevel_3_4, 555
 - UART_FIFOLevel_7_8, 555
 - UART_FIFOLevel_TypeDef, 555
 - UART_Flag_Busy, 556
 - UART_Flag_InvCTS, 556
 - UART_Flag_InvDCD, 556
 - UART_Flag_InvDSR, 556
 - UART_Flag_InvRI, 556
 - UART_Flag_RxFIFOEmpty, 556
 - UART_Flag_RxFIFOFull, 556

- UART_Flag_TxFIFOEmpty, 556
- UART_Flag_TxFIFOFull, 556
- UART_Flag_Typedef, 555
- UART_FlagStatus, 559
- UART_ITCmd, 560
- UART_ITFIFOLevelConfig, 560
- UART_ITMaskedStatus, 560
- UART_ITRawStatus, 562
- UART_ITSource_ChangeCTS, 556
- UART_ITSource_ChangeDCD, 556
- UART_ITSource_ChangeDSR, 556
- UART_ITSource_ChangeRI, 556
- UART_ITSource_ErrorBreak, 556
- UART_ITSource_ErrorFrame, 556
- UART_ITSource_ErrorOverflow, 556
- UART_ITSource_ErrorParity, 556
- UART_ITSource_RecieveTimeout, 556
- UART_ITSource_RxFIFOLevel, 556
- UART_ITSource_TxFIFOLevel, 556
- UART_ITSource_Typedef, 556
- UART_ITStatusClear, 562
- UART_Init, 559
- UART_ModemConfig, 562
- UART_ModemStructInit, 563
- UART_ParityBit_Disable, 556
- UART_ParityBit_Even, 556
- UART_ParityBit_High, 556
- UART_ParityBit_Low, 556
- UART_ParityBit_Odd, 556
- UART_ParityBit_TypeDef, 556
- UART_RecieveData, 563
- UART_SendData, 563
- UART_StopBit_1, 557
- UART_StopBit_2, 557
- UART_StopBit_TypeDef, 556
- UART_StructInit, 564
- niietcm4_userflash.c, 564
 - USERFLASH_FullErase, 565
 - USERFLASH_ITCmd, 567
 - USERFLASH_Info_PageErase, 566
 - USERFLASH_Info_Read, 566
 - USERFLASH_Info_Write, 566
 - USERFLASH_Init, 566
 - USERFLASH_OperationStatus, 567
 - USERFLASH_OperationStatusClear, 567
 - USERFLASH_PageErase, 567
 - USERFLASH_Read, 568
 - USERFLASH_Write, 568
- niietcm4_userflash.h, 568
 - IS_USERFLASH_STATUS, 570
 - USERFLASH_FullErase, 572
 - USERFLASH_INFO_PAGE_SIZE_BYTES↔ES, 571
 - USERFLASH_INFO_PAGE_TOTAL, 571
 - USERFLASH_INFO_TOTAL_BYTES, 571
 - USERFLASH_ITCmd, 573
 - USERFLASH_Info_PageErase, 572
 - USERFLASH_Info_Read, 572
 - USERFLASH_Info_Write, 572
 - USERFLASH_Init, 573
 - USERFLASH_OperationStatus, 573
 - USERFLASH_OperationStatusClear, 573
 - USERFLASH_PAGE_SIZE_BYTES, 571
 - USERFLASH_PAGE_TOTAL, 571
 - USERFLASH_PageErase, 574
 - USERFLASH_Read, 574
 - USERFLASH_Status_Complete, 571
 - USERFLASH_Status_Error, 572
 - USERFLASH_Status_None, 571
 - USERFLASH_Status_TypeDef, 571
 - USERFLASH_TOTAL_BYTES, 571
 - USERFLASH_Write, 574
- PRIVELGED
 - DMA_Protect_TypeDef, 342
- PRM_DATA
 - DMA_ConfigData_TypeDef, 338
- R_POWER
 - _CHANNEL_CFG_bits, 328
- RCC, 278
 - RCC_ADCClk_0
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_1
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_10
 - Типы, 145
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_11
 - Типы, 145
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_2
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_3
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_4
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_5
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_6
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_7
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_8
 - Типы, 144
 - niietcm4_rcc.h, 511
 - RCC_ADCClk_9
 - Типы, 144
 - niietcm4_rcc.h, 511

- RCC_ADCClk_TypeDef
 - Типы, [144](#)
 - niietcm4_rcc.h, [511](#)
- RCC_ADCClkCmd
 - Приватные функции, [283](#)
 - Тактирование ADC, [162](#)
 - niietcm4_rcc.c, [496](#)
 - niietcm4_rcc.h, [515](#)
- RCC_ADCClkDivConfig
 - Приватные функции, [283](#)
 - Тактирование ADC, [162](#)
 - niietcm4_rcc.c, [496](#)
 - niietcm4_rcc.h, [516](#)
- RCC_CLK_CHANGE_TIMEOUT
 - Константы, [139](#)
 - niietcm4_rcc.h, [511](#)
- RCC_PLL_CTRL_Reset_Value
 - Начальные значения регистров, [281](#)
 - niietcm4_rcc.c, [495](#)
- RCC_PLL_NF_Reset_Value
 - Начальные значения регистров, [281](#)
 - niietcm4_rcc.c, [495](#)
- RCC_PLL_NR_Reset_Value
 - Начальные значения регистров, [281](#)
 - niietcm4_rcc.c, [495](#)
- RCC_PLL_OD_Reset_Value
 - Начальные значения регистров, [281](#)
 - niietcm4_rcc.c, [495](#)
- RCC_PLLAutoConfig
 - Конфигурация PLL, [151](#)
 - Приватные функции, [285](#)
 - niietcm4_rcc.c, [497](#)
 - niietcm4_rcc.h, [517](#)
- RCC_PLLDeInit
 - Конфигурация PLL, [152](#)
 - Приватные функции, [285](#)
 - niietcm4_rcc.c, [498](#)
 - niietcm4_rcc.h, [518](#)
- RCC_PLLDiv
 - RCC_PLLInit_TypeDef, [346](#)
- RCC_PLLInit
 - Конфигурация PLL, [152](#)
 - Приватные функции, [285](#)
 - niietcm4_rcc.c, [498](#)
 - niietcm4_rcc.h, [518](#)
- RCC_PLLInit_TypeDef, [345](#)
 - RCC_PLLDiv, [346](#)
 - RCC_PLLNF, [346](#)
 - RCC_PLLNO, [346](#)
 - RCC_PLLNR, [346](#)
 - RCC_PLLRef, [346](#)
- RCC_PLLNF
 - RCC_PLLInit_TypeDef, [346](#)
- RCC_PLLNO
 - RCC_PLLInit_TypeDef, [346](#)
- RCC_PLLNO_Disable
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLNO_Div2
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLNO_Div4
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLNO_TypeDef
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLNR
 - RCC_PLLInit_TypeDef, [346](#)
- RCC_PLLPowerDownCmd
 - Конфигурация PLL, [153](#)
 - Приватные функции, [286](#)
 - niietcm4_rcc.c, [499](#)
 - niietcm4_rcc.h, [519](#)
- RCC_PLLRef
 - RCC_PLLInit_TypeDef, [346](#)
- RCC_PLLRef_ETH_25MHz
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLRef_TypeDef
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLRef_USB_60MHz
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLRef_USB_CLK
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLRef_XI_OSC
 - Типы, [146](#)
 - niietcm4_rcc.h, [513](#)
- RCC_PLLStructInit
 - Конфигурация PLL, [153](#)
 - Приватные функции, [286](#)
 - niietcm4_rcc.c, [499](#)
 - niietcm4_rcc.h, [519](#)
- RCC_PeriphClk_ADC
 - Типы, [145](#)
 - niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_CMP
 - Типы, [145](#)
 - niietcm4_rcc.h, [511](#)
- RCC_PeriphClk_I2C0
 - Типы, [145](#)
 - niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_I2C1
 - Типы, [145](#)
 - niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM0
 - Типы, [145](#)
 - niietcm4_rcc.h, [511](#)
- RCC_PeriphClk_PWM1
 - Типы, [145](#)
 - niietcm4_rcc.h, [511](#)
- RCC_PeriphClk_PWM2
 - Типы, [145](#)

- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM3
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM4
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM5
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM6
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM7
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_PWM8
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClk_QEP0
 - Типы, [145](#)
- niietcm4_rcc.h, [511](#)
- RCC_PeriphClk_QEP1
 - Типы, [145](#)
- niietcm4_rcc.h, [511](#)
- RCC_PeriphClk_TypeDef
 - Типы, [145](#)
- niietcm4_rcc.h, [511](#)
- RCC_PeriphClk_WD
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphClkCmd
 - Приватные функции, [284](#)
 - Управление тактированием, [154](#)
- niietcm4_rcc.c, [496](#)
 - niietcm4_rcc.h, [516](#)
- RCC_PeriphRst_CAP0
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_CAP1
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_CAP2
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_CAP3
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_CAP4
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_CAP5
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_CMP
 - Типы, [146](#)
- niietcm4_rcc.h, [513](#)
- RCC_PeriphRst_ETH
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_I2C0
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_I2C1
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM0
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM1
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM2
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM3
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM4
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM5
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM6
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM7
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_PWM8
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_QEP0
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_QEP1
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_SPI0
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_SPI1
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_SPI2
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_SPI3
 - Типы, [146](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_Timer0
 - Типы, [145](#)
- niietcm4_rcc.h, [512](#)
- RCC_PeriphRst_Timer1
 - Типы, [145](#)

- niietcm4_rcc.h, 512
- RCC_PeriphRst_Timer2
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_TypeDef
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_UART0
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_UART1
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_UART2
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_UART3
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_USB
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRst_WD
 - Типы, 145
 - niietcm4_rcc.h, 512
- RCC_PeriphRstCmd
 - Приватные функции, 284
 - Управление сбросом, 164
 - niietcm4_rcc.c, 497
 - niietcm4_rcc.h, 517
- RCC_SPIClk_SYSCLK
 - Типы, 147
 - niietcm4_rcc.h, 513
- RCC_SPIClk_TypeDef
 - Типы, 146
 - niietcm4_rcc.h, 513
- RCC_SPIClk_USB_60MHz
 - Типы, 147
 - niietcm4_rcc.h, 513
- RCC_SPIClk_USB_CLK
 - Типы, 147
 - niietcm4_rcc.h, 513
- RCC_SPIClk_XI_OSC
 - Типы, 147
 - niietcm4_rcc.h, 513
- RCC_SPIClkCmd
 - Приватные функции, 287
 - Тактирование SPI, 160
 - niietcm4_rcc.c, 499
 - niietcm4_rcc.h, 519
- RCC_SPIClkDivConfig
 - Приватные функции, 287
 - Тактирование SPI, 160
 - niietcm4_rcc.c, 501
 - niietcm4_rcc.h, 520
- RCC_SPIClkSel
 - Приватные функции, 287
 - Тактирование SPI, 161
- niietcm4_rcc.c, 501
 - niietcm4_rcc.h, 520
- RCC_SysClk_CPE_Sel
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_ETH25MHz
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_PLL
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_PLLDIV
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_POR
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_TypeDef
 - Типы, 147
 - niietcm4_rcc.h, 513
- RCC_SysClk_USB60MHz
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_USB_CLK
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClk_XI_OSC
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_SysClkDiv2Out
 - Функции, 149
 - Приватные функции, 288
 - niietcm4_rcc.c, 501
 - niietcm4_rcc.h, 520
- RCC_SysClkSel
 - Приватные функции, 288
 - Управление тактированием, 155
 - niietcm4_rcc.c, 502
 - niietcm4_rcc.h, 521
- RCC_SysClkStatus
 - Приватные функции, 288
 - Управление тактированием, 155
 - niietcm4_rcc.c, 502
 - niietcm4_rcc.h, 521
- RCC_UARTClk_SYSCLK
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_UARTClk_TypeDef
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_UARTClk_USB_60MHz
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_UARTClk_USB_CLK
 - Типы, 147
 - niietcm4_rcc.h, 514
- RCC_UARTClk_XI_OSC
 - Типы, 147

- niietcm4_rcc.h, [514](#)
- RCC_UARTClkCmd
 - Приватные функции, [289](#)
 - Тактирование UART, [158](#)
- niietcm4_rcc.c, [502](#)
 - niietcm4_rcc.h, [521](#)
- RCC_UARTClkDivConfig
 - Приватные функции, [289](#)
 - Тактирование UART, [158](#)
- niietcm4_rcc.c, [503](#)
 - niietcm4_rcc.h, [521](#)
- RCC_UARTClkSel
 - Приватные функции, [289](#)
 - Тактирование UART, [159](#)
- niietcm4_rcc.c, [503](#)
 - niietcm4_rcc.h, [522](#)
- RCC_USBClk_TypeDef
 - Типы, [147](#)
- niietcm4_rcc.h, [514](#)
- RCC_USBClk_USB_CLK
 - Типы, [147](#)
- niietcm4_rcc.h, [514](#)
- RCC_USBClk_XI_OSC
 - Типы, [147](#)
- niietcm4_rcc.h, [514](#)
- RCC_USBClkCmd
 - Приватные функции, [289](#)
 - Тактирование USB, [156](#)
- niietcm4_rcc.c, [503](#)
 - niietcm4_rcc.h, [522](#)
- RCC_USBClkConfig
 - Приватные функции, [290](#)
 - Тактирование USB, [156](#)
- niietcm4_rcc.c, [504](#)
 - niietcm4_rcc.h, [522](#)
- RCC_USBFreq_12MHz
 - Типы, [148](#)
- niietcm4_rcc.h, [514](#)
- RCC_USBFreq_24MHz
 - Типы, [148](#)
- niietcm4_rcc.h, [514](#)
- RCC_USBFreq_TypeDef
 - Типы, [147](#)
- niietcm4_rcc.h, [514](#)
- RCC_WaitClkChange
 - Приватные функции, [290](#)
- niietcm4_rcc.c, [504](#)
- RESERVED0
 - DMA_ConfigData_TypeDef, [339](#)
- RESERVED1
 - DMA_ConfigData_TypeDef, [339](#)
- RTC, [291](#)
- RTC_Date_TypeDef, [347](#)
 - RTC_Day, [347](#)
 - RTC_Month, [347](#)
 - RTC_Weekday, [347](#)
 - RTC_Year, [347](#)
- RTC_Day
 - RTC_Date_TypeDef, [347](#)
- RTC_Date_TypeDef, [347](#)
- RTC_Format_BCD
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Format_BIN
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Format_TypeDef
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Hour
 - RTC_Time_TypeDef, [348](#)
- RTC_Minute
 - RTC_Time_TypeDef, [348](#)
- RTC_Month
 - RTC_Date_TypeDef, [347](#)
- RTC_Month_April
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_August
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_December
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_February
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_January
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_July
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_June
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_March
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_May
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_November
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_October
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_September
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Month_TypeDef
 - Типы, [167](#)
- niietcm4_rtc.h, [527](#)
- RTC_Psecond
 - RTC_Time_TypeDef, [348](#)
- RTC_Second
 - RTC_Time_TypeDef, [348](#)

- RTC_Time_TypeDef, 348
- RTC_Time_TypeDef, 348
 - RTC_Hour, 348
 - RTC_Minute, 348
 - RTC_Psecond, 348
 - RTC_Second, 348
- RTC_Weekday
 - RTC_Date_TypeDef, 347
- RTC_Weekday_Friday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Weekday_Monday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Weekday_Saturday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Weekday_Sunday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Weekday_Thursday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Weekday_Tuesday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Weekday_TypeDef
 - Типы, 167
 - niietcm4_rtc.h, 527
- RTC_Weekday_Wednesday
 - Типы, 167
 - niietcm4_rtc.h, 528
- RTC_Year
 - RTC_Date_TypeDef, 347
- SRC_DATA_END
 - DMA_Channel_TypeDef, 335
- SRC_INC
 - _CHANNEL_CFG_bits, 329
- SRC_PROT_BUFFERABLE
 - _CHANNEL_CFG_bits, 329
- SRC_PROT_CACHEABLE
 - _CHANNEL_CFG_bits, 329
- SRC_PROT_PRIVILEGED
 - _CHANNEL_CFG_bits, 329
- SRC_SIZE
 - _CHANNEL_CFG_bits, 329
- TIMER, 294
- TIMER_Cmd
 - Конфигурация, 173
 - Приватные функции, 297
 - niietcm4_timer.c, 529
 - niietcm4_timer.h, 535
- TIMER_ExtInput_CountClk
 - Типы, 169
 - niietcm4_timer.h, 535
- TIMER_ExtInput_CountEn
 - Типы, 169
- niietcm4_timer.h, 535
- TIMER_ExtInput_Disable
 - Типы, 169
- niietcm4_timer.h, 535
- TIMER_ExtInput_TypeDef
 - Типы, 169
- niietcm4_timer.h, 535
- TIMER_ExtInputConfig
 - Конфигурация, 174
 - Приватные функции, 298
 - niietcm4_timer.c, 530
 - niietcm4_timer.h, 535
- TIMER_FreqConfig
 - Конфигурация, 174
 - Приватные функции, 298
 - niietcm4_timer.c, 530
 - niietcm4_timer.h, 536
- TIMER_GetCounter
 - Конфигурация, 174
 - Приватные функции, 298
 - niietcm4_timer.c, 530
 - niietcm4_timer.h, 536
- TIMER_GetReload
 - Конфигурация, 175
 - Приватные функции, 299
 - niietcm4_timer.c, 531
 - niietcm4_timer.h, 536
- TIMER_ITCmd
 - Прерывания, 177
 - Приватные функции, 299
 - niietcm4_timer.c, 531
 - niietcm4_timer.h, 537
- TIMER_ITStatus
 - Прерывания, 177
 - Приватные функции, 299
 - niietcm4_timer.c, 531
 - niietcm4_timer.h, 537
- TIMER_ITStatusClear
 - Прерывания, 178
 - Приватные функции, 299
 - niietcm4_timer.c, 531
 - niietcm4_timer.h, 537
- TIMER_PeriodConfig
 - Конфигурация, 175
 - Приватные функции, 300
 - niietcm4_timer.c, 532
 - niietcm4_timer.h, 537
- TIMER_SetCounter
 - Конфигурация, 175
 - Приватные функции, 300
 - niietcm4_timer.c, 532
 - niietcm4_timer.h, 538
- TIMER_SetReload
 - Конфигурация, 176
 - Приватные функции, 300
 - niietcm4_timer.c, 532
 - niietcm4_timer.h, 538
- UART, 302

- UART_BaudRate
 - UART_Init_TypeDef, 349
- UART_BaudRateDivConfig
 - Функции, 187
 - Приватные функции, 306
 - niietcm4_uart.c, 541
 - niietcm4_uart.h, 557
- UART_Break
 - Функции, 188
 - Приватные функции, 306
 - niietcm4_uart.c, 542
 - niietcm4_uart.h, 557
- UART_CTSEn
 - UART_ModemInit_TypeDef, 351
- UART_ClkFreq
 - UART_Init_TypeDef, 349
- UART_Cmd
 - Функции, 188
 - Приватные функции, 307
 - niietcm4_uart.c, 542
 - niietcm4_uart.h, 557
- UART_DMABlkOnErrCmd
 - Настройка DMA, 198
 - Приватные функции, 307
 - niietcm4_uart.c, 543
 - niietcm4_uart.h, 558
- UART_DMACmd
 - Настройка DMA, 198
 - Приватные функции, 307
 - niietcm4_uart.c, 543
 - niietcm4_uart.h, 558
- UART_DataWidth
 - UART_Init_TypeDef, 349
- UART_DataWidth_5
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_DataWidth_6
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_DataWidth_7
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_DataWidth_8
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_DataWidth_TypeDef
 - Типы, 182
 - niietcm4_uart.h, 554
- UART_DeInit
 - Инициализация и деинициализация, 189
 - Приватные функции, 307
 - niietcm4_uart.c, 542
 - niietcm4_uart.h, 558
- UART_Dir_Rx
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_Dir_Tx
 - Типы, 183
- niietcm4_uart.h, 555
- UART_Dir_Typedef
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_Error_Break
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_Error_Frame
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_Error_Overflow
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_Error_Parity
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_Error_Typedef
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_ErrorStatus
 - Прием и передача, 191
 - Приватные функции, 308
 - niietcm4_uart.c, 543
 - niietcm4_uart.h, 558
- UART_ErrorStatusClear
 - Прием и передача, 191
 - Приватные функции, 308
 - niietcm4_uart.c, 543
 - niietcm4_uart.h, 559
- UART_FIFOEn
 - UART_Init_TypeDef, 349
- UART_FIFOLevel_1_2
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FIFOLevel_1_4
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FIFOLevel_1_8
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FIFOLevel_3_4
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FIFOLevel_7_8
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FIFOLevel_TypeDef
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FIFOLevelRx
 - UART_Init_TypeDef, 350
- UART_FIFOLevelTx
 - UART_Init_TypeDef, 350
- UART_Flag_Busy
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_InvCTS
 - Типы, 184

- niietcm4_uart.h, 556
- UART_Flag_InvDCD
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_InvDSR
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_InvRI
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_RxFIFOEmpty
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_RxFIFOFull
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_TxFIFOEmpty
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_TxFIFOFull
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_Flag_Typedef
 - Типы, 183
 - niietcm4_uart.h, 555
- UART_FlagStatus
 - Прием и передача, 192
 - Приватные функции, 308
 - niietcm4_uart.c, 545
 - niietcm4_uart.h, 559
- UART_ITCmd
 - Прерывания, 195
 - Приватные функции, 309
 - niietcm4_uart.c, 545
 - niietcm4_uart.h, 560
- UART_ITFIFOLevelConfig
 - Прерывания, 196
 - Приватные функции, 309
 - niietcm4_uart.c, 546
 - niietcm4_uart.h, 560
- UART_ITMaskedStatus
 - Прерывания, 196
 - Приватные функции, 310
 - niietcm4_uart.c, 546
 - niietcm4_uart.h, 560
- UART_ITRawStatus
 - Прерывания, 196
 - Приватные функции, 310
 - niietcm4_uart.c, 546
 - niietcm4_uart.h, 562
- UART_ITSource_ChangeCTS
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_ChangeDCD
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_ChangeDSR
 - Типы, 184
- niietcm4_uart.h, 556
- UART_ITSource_ChangeRI
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_ErrorBreak
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_ErrorFrame
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_ErrorOverflow
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_ErrorParity
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_RecieveTimeout
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_RxFIFOLevel
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_TxFIFOLevel
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITSource_Typedef
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ITStatusClear
 - Прерывания, 197
 - Приватные функции, 310
 - niietcm4_uart.c, 547
 - niietcm4_uart.h, 562
- UART_Init
 - Инициализация и деинициализация, 189
 - Приватные функции, 309
 - niietcm4_uart.c, 545
 - niietcm4_uart.h, 559
- UART_Init_TypeDef, 349
 - UART_BaudRate, 349
 - UART_ClkFreq, 349
 - UART_DataWidth, 349
 - UART_FIFOEn, 349
 - UART_FIFOLevelRx, 350
 - UART_FIFOLevelTx, 350
 - UART_ParityBit, 350
 - UART_RxEn, 350
 - UART_StopBit, 350
 - UART_TxEn, 350
- UART_InvDTR
 - UART_ModemInit_TypeDef, 351
- UART_InvRTS
 - UART_ModemInit_TypeDef, 351
- UART_ModemConfig
 - Приватные функции, 311
 - Режим модема, 193
 - niietcm4_uart.c, 547
 - niietcm4_uart.h, 562

- UART_ModemInit_TypeDef, 351
 - UART_CTSEn, 351
 - UART_InvDTR, 351
 - UART_InvRTS, 351
 - UART_RTSEn, 351
- UART_ModemStructInit
 - Приватные функции, 311
 - Режим модема, 193
 - niietcm4_uart.c, 547
 - niietcm4_uart.h, 563
- UART_ParityBit
 - UART_Init_TypeDef, 350
- UART_ParityBit_Disable
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ParityBit_Even
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ParityBit_High
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ParityBit_Low
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ParityBit_Odd
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_ParityBit_TypeDef
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_RTSEn
 - UART_ModemInit_TypeDef, 351
- UART_RecieveData
 - Прием и передача, 192
 - Приватные функции, 311
 - niietcm4_uart.c, 548
 - niietcm4_uart.h, 563
- UART_RxEn
 - UART_Init_TypeDef, 350
- UART_SendData
 - Прием и передача, 192
 - Приватные функции, 312
 - niietcm4_uart.c, 548
 - niietcm4_uart.h, 563
- UART_StopBit
 - UART_Init_TypeDef, 350
- UART_StopBit_1
 - Типы, 185
 - niietcm4_uart.h, 557
- UART_StopBit_2
 - Типы, 185
 - niietcm4_uart.h, 557
- UART_StopBit_TypeDef
 - Типы, 184
 - niietcm4_uart.h, 556
- UART_StructInit
 - Инициализация и деинициализация, 190
 - Приватные функции, 312
 - niietcm4_uart.c, 548
 - niietcm4_uart.h, 564
- UART_TxEn
 - UART_Init_TypeDef, 350
- USERFLASH, 313
- USERFLASH_FullErase
 - Основная область флеш, 206
 - Приватные функции, 316
 - niietcm4_userflash.c, 565
 - niietcm4_userflash.h, 572
- USERFLASH_INFO_PAGE_SIZE_BYTES
 - Информационная область флеш, 202
 - niietcm4_userflash.h, 571
- USERFLASH_INFO_PAGE_TOTAL
 - Информационная область флеш, 202
 - niietcm4_userflash.h, 571
- USERFLASH_INFO_TOTAL_BYTES
 - Информационная область флеш, 202
 - niietcm4_userflash.h, 571
- USERFLASH_ITCmd
 - Функции, 204
 - Приватные функции, 318
 - niietcm4_userflash.c, 567
 - niietcm4_userflash.h, 573
- USERFLASH_Info_PageErase
 - Информационная область флеш, 208
 - Приватные функции, 317
 - niietcm4_userflash.c, 566
 - niietcm4_userflash.h, 572
- USERFLASH_Info_Read
 - Информационная область флеш, 208
 - Приватные функции, 317
 - niietcm4_userflash.c, 566
 - niietcm4_userflash.h, 572
- USERFLASH_Info_Write
 - Информационная область флеш, 209
 - Приватные функции, 317
 - niietcm4_userflash.c, 566
 - niietcm4_userflash.h, 572
- USERFLASH_Init
 - Функции, 204
 - Приватные функции, 317
 - niietcm4_userflash.c, 566
 - niietcm4_userflash.h, 573
- USERFLASH_OperationStatus
 - Функции, 205
 - Приватные функции, 318
 - niietcm4_userflash.c, 567
 - niietcm4_userflash.h, 573
- USERFLASH_OperationStatusClear
 - Функции, 205
 - Приватные функции, 318
 - niietcm4_userflash.c, 567
 - niietcm4_userflash.h, 573
- USERFLASH_PAGE_SIZE_BYTES
 - Основная область флеш, 201
 - niietcm4_userflash.h, 571
- USERFLASH_PAGE_TOTAL

- Основная область флеш, [201](#)
- [niietcm4_userflash.h](#), [571](#)
- USERFLASH_PageErase
 - Основная область флеш, [206](#)
 - Приватные функции, [318](#)
 - [niietcm4_userflash.c](#), [567](#)
 - [niietcm4_userflash.h](#), [574](#)
- USERFLASH_Read
 - Основная область флеш, [207](#)
 - Приватные функции, [319](#)
 - [niietcm4_userflash.c](#), [568](#)
 - [niietcm4_userflash.h](#), [574](#)
- USERFLASH_Status_Complete
 - Типы, [203](#)
 - [niietcm4_userflash.h](#), [571](#)
- USERFLASH_Status_Error
 - Типы, [203](#)
 - [niietcm4_userflash.h](#), [572](#)
- USERFLASH_Status_None
 - Типы, [203](#)
 - [niietcm4_userflash.h](#), [571](#)
- USERFLASH_Status_TypeDef
 - Типы, [203](#)
 - [niietcm4_userflash.h](#), [571](#)
- USERFLASH_TOTAL_BYTES
 - Основная область флеш, [201](#)
 - [niietcm4_userflash.h](#), [571](#)
- USERFLASH_Write
 - Основная область флеш, [207](#)
 - Приватные функции, [319](#)
 - [niietcm4_userflash.c](#), [568](#)
 - [niietcm4_userflash.h](#), [574](#)