

APTX 1.1 mcu software

1.0

Generated by Doxygen 1.8.16

1 Module Index	1
1.1 Modules	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Module Documentation	7
4.1 CORE	7
4.1.1 Detailed Description	7
4.2 Armenta	8
4.2.1 Detailed Description	8
4.3 Globals	9
4.3.1 Detailed Description	9
4.3.2 Macro Definition Documentation	9
4.3.2.1 ADC_Battery	9
4.3.2.2 ADC_Pressure	10
4.3.2.3 ADC_Pressure_20_bar	10
4.3.2.4 ADC_Pressure_27_bar	10
4.3.2.5 APP_ADDRESS	10
4.3.2.6 APP_ADDRESS_P	10
4.3.2.7 APP_SIZE	11
4.3.2.8 DEBUG_PRINT	11
4.3.2.9 DONT_BEEP	11
4.3.2.10 END_ADDRESS	11
4.3.2.11 EXPERIMENTAL	11
4.3.2.12 FROM_BOOTLOADER	12
4.3.2.13 MAX_HOLE_SIZE	12
4.3.2.14 MCU_VERSION_MAJOR	12
4.3.2.15 MCU_VERSION_MINOR	12
4.3.2.16 MCU_VERSION_PATCH	12
4.3.2.17 MCU_VERSION_RC	13
4.3.2.18 PASS_PRESSURE_CHECK	13
4.3.2.19 PASS_SPI	13
4.3.2.20 TECHNICIAN_ENABLED	13
4.3.2.21 VERSION	13
4.4 SPI	14
4.4.1 Detailed Description	14
4.4.2 Macro Definition Documentation	14
4.4.2.1 BLOCK_ERASE_32K	14
4.4.2.2 BLOCK_ERASE_4K	15

4.4.2.3 BLOCK_ERASE_64K	15
4.4.2.4 BYTE_PROGRAM	15
4.4.2.5 ERASE_CHIP1	15
4.4.2.6 ERASE_CHIP2	15
4.4.2.7 HIGH	15
4.4.2.8 LOW	16
4.4.2.9 READ_ARRAY1	16
4.4.2.10 READ_ARRAY2	16
4.4.2.11 READ_DEVICE_ID	16
4.4.2.12 READ_DUAL	16
4.4.2.13 READ_MANUFACTURE_ID	16
4.4.2.14 READ_QUAD	17
4.4.2.15 READ_STATUS_REGISTER1	17
4.4.2.16 READ_STATUS_REGISTER2	17
4.4.2.17 WRITE_DISABLE	17
4.4.2.18 WRITE_ENABLE	17
4.4.2.19 WRITE_ENABLE_STATUS_REGISTER_VOLATILE	17
4.4.2.20 WRITE_STATUS_REGISTER	17
4.5 Auxiliary	18
4.5.1 Detailed Description	18
4.5.2 Macro Definition Documentation	18
4.5.2.1 AM_WARNING_TIMES	18
4.5.2.2 BEEP_COUNTER_0	19
4.5.2.3 BEEP_COUNTER_200	19
4.5.2.4 BEEP_COUNTER_400	19
4.5.2.5 Beep_Every_X_Counting	19
4.5.2.6 Configuration_register_add	19
4.5.2.7 HDC_1080_ADD	19
4.5.2.8 Humidity_register_add	20
4.5.2.9 KEY_PRESS	20
4.5.2.10 MOTOR_1_PWM_OFF	20
4.5.2.11 MOTOR_1_PWM_ON	20
4.5.2.12 MOTOR_1_PWM_ON_50	20
4.5.2.13 MOTOR_1_PWM_ON_75	20
4.5.2.14 RXBUFFERSIZE	21
4.5.2.15 SERIAL_PC	21
4.5.2.16 SERIAL_SCREEN	21
4.5.2.17 START_BEEP	21
4.5.2.18 Temperature_register_add	21
4.5.2.19 TICKS_FOR_UPDATE	21
4.5.2.20 TICKS_SCREEN_UPDATE_LONG	22
4.5.2.21 TICKS_SCREEN_UPDATE_SUPER_LONG	22

4.5.2.22 TIMEOUT_SCREEN	22
4.6 ADC	23
4.6.1 Detailed Description	23
4.6.2 Macro Definition Documentation	23
4.6.2.1 ADC_RESERVED	23
4.6.2.2 ADC_RESERVED2	24
4.6.2.3 FIFOREAD	24
4.6.2.4 HARDWARE_DEFAULT	24
4.6.2.5 SELECT_CHANNEL_0	24
4.6.2.6 SELECT_CHANNEL_1	24
4.6.2.7 SELECT_CHANNEL_2	24
4.6.2.8 SELECT_CHANNEL_3	25
4.6.2.9 SELECT_CHANNEL_4	25
4.6.2.10 SELECT_CHANNEL_5	25
4.6.2.11 SELECT_CHANNEL_6	25
4.6.2.12 SELECT_CHANNEL_7	25
4.6.2.13 SELECT_TEST_REFM	25
4.6.2.14 SELECT_TEST_REFP	26
4.6.2.15 SELECT_TEST_VDIV2	26
4.6.2.16 WRITE_CFR	26
4.7 STM32L0xx_IAP_Main	27
4.7.1 Detailed Description	27
4.7.2 Function Documentation	27
4.7.2.1 Int2Str()	27
4.7.2.2 Serial_PutByte()	28
4.7.2.3 Serial_PutString()	28
4.7.2.4 Str2Int()	29
4.8 STM32L0xx_IAP	31
4.8.1 Detailed Description	31
4.8.2 Function Documentation	31
4.8.2.1 dump_mem()	31
4.8.2.2 FLASH_If_Erase()	32
4.8.2.3 FLASH_If_Write()	32
4.8.2.4 Main_Menu()	33
4.8.3 Variable Documentation	34
4.8.3.1 aFileName	35
4.8.3.2 FlashProtection	35
4.8.3.3 JumpAddress	35
4.8.3.4 JumpToApplication	35
4.8.3.5 memory_extern [1/2]	35
4.8.3.6 memory_extern [2/2]	35
4.9 CMSIS	36

4.9.1 Detailed Description	36
4.10 Stm32f0xx_system	37
4.10.1 Detailed Description	37
4.11 STM32F0xx_System_Private_Includes	38
4.12 STM32F0xx_System_Private_TypesDefinitions	39
4.13 STM32F0xx_System_Private_Defines	40
4.13.1 Detailed Description	40
4.13.2 Macro Definition Documentation	40
4.13.2.1 HSE_VALUE	40
4.13.2.2 HSI48_VALUE	40
4.13.2.3 HSI_VALUE	40
4.14 STM32F0xx_System_Private_Macros	41
4.15 STM32F0xx_System_Private_Variables	42
4.15.1 Detailed Description	42
4.15.2 Variable Documentation	42
4.15.2.1 AHBPrescTable	42
4.15.2.2 APBPrescTable	42
4.15.2.3 SystemCoreClock	42
4.16 STM32F0xx_System_Private_FunctionPrototypes	43
4.17 STM32F0xx_System_Private_Functions	44
4.17.1 Detailed Description	44
4.17.2 Function Documentation	44
4.17.2.1 SystemCoreClockUpdate()	44
4.17.2.2 SystemInit()	45
5 Class Documentation	47
5.1 adc_command Union Reference	47
5.1.1 Detailed Description	48
5.1.2 Member Data Documentation	48
5.1.2.1 b1	48
5.1.2.2 b2	48
5.1.2.3 bits	48
5.1.2.4 bytes	49
5.1.2.5 CLK_SOURCE	49
5.1.2.6 CMR	49
5.1.2.7 CONV_OUTPUT_FORMAT	49
5.1.2.8 CONVERSION_MODE_SELECT	49
5.1.2.9 D11	49
5.1.2.10 EOC_INT_FUNC	50
5.1.2.11 FIFO_TRIG_LEVEL	50
5.1.2.12 INPUT_SELECT_MODE	50
5.1.2.13 SAMPLE_PERIOD	50

5.1.2.14 SWEEP_SEQ_SELECT	50
5.1.2.15 value	50
5.2 AdcController Class Reference	51
5.2.1 Detailed Description	52
5.2.2 Constructor & Destructor Documentation	52
5.2.2.1 AdcController()	52
5.2.2.2 ~AdcController()	52
5.2.3 Member Function Documentation	53
5.2.3.1 convert_to_volt()	53
5.2.3.2 pressure_check()	53
5.2.3.3 read() [1/2]	54
5.2.3.4 read() [2/2]	55
5.2.4 Member Data Documentation	55
5.2.4.1 bat_percentage	55
5.2.4.2 battery_volt	55
5.2.4.3 channels_buffer	55
5.2.4.4 cs_register	55
5.2.4.5 current_sense	56
5.2.4.6 dma_ring_buffer	56
5.2.4.7 pressure_ok	56
5.2.4.8 stall_pressure	56
5.2.4.9 temperature_mcu	56
5.2.4.10 vbat	56
5.2.4.11 vref	57
5.3 bypass_state Union Reference	57
5.3.1 Detailed Description	57
5.3.2 Member Data Documentation	58
5.3.2.1 bits	58
5.3.2.2 bypass1	58
5.3.2.3 bypass2	58
5.3.2.4 bypass3	58
5.3.2.5 bypass4	58
5.3.2.6 reserved	58
5.3.2.7 value	59
5.4 byte8_t_mem_status Union Reference	59
5.4.1 Detailed Description	60
5.4.2 Member Data Documentation	60
5.4.2.1 bits	60
5.4.2.2 BP0	60
5.4.2.3 BP1	61
5.4.2.4 BP2	61
5.4.2.5 bytes	61

5.4.2.6 CMP	61
5.4.2.7 LB1	61
5.4.2.8 LB2	61
5.4.2.9 LB3	62
5.4.2.10 QE	62
5.4.2.11 RES	62
5.4.2.12 RES2	62
5.4.2.13 SEC	62
5.4.2.14 SRP0	62
5.4.2.15 SRP1	63
5.4.2.16 status1	63
5.4.2.17 status2	63
5.4.2.18 TB	63
5.4.2.19 value	63
5.4.2.20 WEL	63
5.4.2.21 WIP	64
5.5 byte8_t_reg_cs Union Reference	64
5.5.1 Detailed Description	65
5.5.2 Member Data Documentation	65
5.5.2.1 bits	65
5.5.2.2 DECODER	65
5.5.2.3 INA_MOTOR	65
5.5.2.4 INB_MOTOR	65
5.5.2.5 nibble_CS	65
5.5.2.6 nibble_PERIPH	66
5.5.2.7 nibbles	66
5.5.2.8 QC	66
5.5.2.9 QG	66
5.5.2.10 QH	66
5.5.2.11 value	66
5.6 byte8_t_reg_readback Union Reference	67
5.6.1 Detailed Description	67
5.6.2 Member Data Documentation	67
5.6.2.1 bits	67
5.6.2.2 CS_MEM2	68
5.6.2.3 CS_MEM3	68
5.6.2.4 CS_STATUS_INV	68
5.6.2.5 DECODER	68
5.6.2.6 INA_MOTOR	68
5.6.2.7 INB_MOTOR	68
5.6.2.8 value	69
5.7 byte8_t_reg_status Union Reference	69

5.7.1 Detailed Description	70
5.7.2 Member Data Documentation	70
5.7.2.1 bits	70
5.7.2.2 FIRE_STATUS	70
5.7.2.3 nibble_OPTO	70
5.7.2.4 nibble_PERIPH	70
5.7.2.5 nibbles	70
5.7.2.6 OPTO_A	71
5.7.2.7 OPTO_B	71
5.7.2.8 OPTO_C	71
5.7.2.9 OPTO_D	71
5.7.2.10 OPTO_STATUS	71
5.7.2.11 RESERVED	71
5.7.2.12 SWITCHES_STATUS	72
5.7.2.13 value	72
5.8 channel Union Reference	72
5.8.1 Detailed Description	73
5.8.2 Member Data Documentation	73
5.8.2.1 b1	73
5.8.2.2 b2	73
5.8.2.3 bytes	73
5.8.2.4 value	73
5.9 CS_reg Class Reference	74
5.9.1 Detailed Description	75
5.9.2 Constructor & Destructor Documentation	75
5.9.2.1 CS_reg()	75
5.9.2.2 ~CS_reg()	75
5.9.3 Member Function Documentation	75
5.9.3.1 cs1()	75
5.9.3.2 cs2()	76
5.9.3.3 cs3()	76
5.9.3.4 cs_a2d()	76
5.9.3.5 cs_clear_opto_counter()	77
5.9.3.6 cs_status()	77
5.9.3.7 motor_cmd()	78
5.9.3.8 print_current()	79
5.9.4 Member Data Documentation	79
5.9.4.1 cs1_value	79
5.9.4.2 current	79
5.9.4.3 readback_register	79
5.10 MainController Class Reference	80
5.10.1 Detailed Description	81

5.10.2 Constructor & Destructor Documentation	81
5.10.2.1 MainController()	82
5.10.2.2 ~MainController()	82
5.10.3 Member Function Documentation	82
5.10.3.1 active_state()	83
5.10.3.2 idle_state()	84
5.10.3.3 init_before_while()	86
5.10.3.4 init_state()	86
5.10.3.5 need_to_stall()	88
5.10.3.6 qa_state()	89
5.10.3.7 uart_state()	90
5.10.3.8 update_state()	91
5.10.4 Member Data Documentation	92
5.10.4.1 adc	92
5.10.4.2 am_sn	93
5.10.4.3 am_valid	93
5.10.4.4 apt_sn	93
5.10.4.5 apt_valid	93
5.10.4.6 battery	93
5.10.4.7 battery_percent	93
5.10.4.8 cursor	94
5.10.4.9 debug	94
5.10.4.10 device	94
5.10.4.11 is_counting_up	94
5.10.4.12 max_count	94
5.10.4.13 mcu_sn	94
5.10.4.14 mcu_valid	95
5.10.4.15 mem	95
5.10.4.16 motor_running	95
5.10.4.17 pressure	95
5.10.4.18 pressure_ok	95
5.10.4.19 pulses	95
5.10.4.20 pulses_on_screen	96
5.10.4.21 screen	96
5.10.4.22 state	96
5.10.4.23 status	96
5.11 Mem_ctrl Class Reference	97
5.11.1 Detailed Description	98
5.11.2 Constructor & Destructor Documentation	98
5.11.2.1 Mem_ctrl()	98
5.11.2.2 ~Mem_ctrl()	99
5.11.3 Member Function Documentation	99

5.11.3.1 <code>binary_and_patch()</code>	99
5.11.3.2 <code>check_connections()</code> [1/2]	100
5.11.3.3 <code>check_connections()</code> [2/2]	100
5.11.3.4 <code>check_connections_with_printback()</code>	101
5.11.3.5 <code>erase()</code> [1/2]	102
5.11.3.6 <code>erase()</code> [2/2]	102
5.11.3.7 <code>get_date()</code>	103
5.11.3.8 <code>get_max()</code>	103
5.11.3.9 <code>get_serial()</code>	104
5.11.3.10 <code>poll_complete()</code>	104
5.11.3.11 <code>print_register()</code>	105
5.11.3.12 <code>read()</code>	106
5.11.3.13 <code>read256()</code>	107
5.11.3.14 <code>write()</code>	108
5.11.3.15 <code>write_register()</code>	109
5.11.4 Member Data Documentation	110
5.11.4.1 <code>AM_MEM</code>	110
5.11.4.2 <code>APT_MEM</code>	110
5.11.4.3 <code>cs_register</code>	110
5.11.4.4 <code>MCU_MEM</code>	110
5.12 Memory Class Reference	111
5.12.1 Detailed Description	112
5.12.2 Constructor & Destructor Documentation	112
5.12.2.1 <code>Memory()</code>	112
5.12.2.2 <code>~Memory()</code>	113
5.12.3 Member Function Documentation	113
5.12.3.1 <code>check_id()</code>	113
5.12.3.2 <code>erase()</code> [1/2]	114
5.12.3.3 <code>erase()</code> [2/2]	114
5.12.3.4 <code>print_status()</code>	114
5.12.3.5 <code>read()</code>	115
5.12.3.6 <code>read256()</code>	116
5.12.3.7 <code>read_status_register1()</code>	117
5.12.3.8 <code>read_status_register2()</code>	117
5.12.3.9 <code>write()</code>	118
5.12.3.10 <code>write_disable()</code>	119
5.12.3.11 <code>write_enable()</code>	119
5.12.3.12 <code>write_status_register()</code>	120
5.12.4 Member Data Documentation	120
5.12.4.1 <code>CS</code>	120
5.12.4.2 <code>cursor</code>	120
5.12.4.3 <code>date</code>	120

5.12.4.4 id	121
5.12.4.5 id_valid	121
5.12.4.6 max_count	121
5.12.4.7 name	121
5.12.4.8 pulses	121
5.12.4.9 serial_number	121
5.12.4.10 status16_t	122
5.13 Monitor Class Reference	122
5.13.1 Detailed Description	124
5.13.2 Constructor & Destructor Documentation	124
5.13.2.1 Monitor()	124
5.13.2.2 ~Monitor()	125
5.13.3 Member Function Documentation	125
5.13.3.1 rx_debug()	125
5.13.3.2 rx_erase()	125
5.13.3.3 rx_exit()	126
5.13.3.4 rx_find_pulses()	126
5.13.3.5 rx_get_man_id()	127
5.13.3.6 rx_go_active()	127
5.13.3.7 rx_help()	128
5.13.3.8 rx_nuke()	129
5.13.3.9 rx_pass()	129
5.13.3.10 rx_read()	130
5.13.3.11 rx_scan() [1/2]	131
5.13.3.12 rx_scan() [2/2]	131
5.13.3.13 rx_set_date()	132
5.13.3.14 rx_set_maxi()	132
5.13.3.15 rx_set_serial()	133
5.13.3.16 rx_start_motor()	134
5.13.3.17 rx_status_read()	134
5.13.3.18 rx_status_write()	135
5.13.3.19 rx_stop_motor()	136
5.13.3.20 rx_test_read()	136
5.13.3.21 rx_upload()	137
5.13.3.22 rx_write()	138
5.13.3.23 serial_consume()	139
5.13.4 Member Data Documentation	141
5.13.4.1 address	141
5.13.4.2 apt	142
5.13.4.3 buffer	142
5.13.4.4 chip_select	142
5.13.4.5 value	142

5.14 PeripheralDevices Class Reference	143
5.14.1 Detailed Description	144
5.14.2 Constructor & Destructor Documentation	144
5.14.2.1 PeripheralDevices()	144
5.14.2.2 ~PeripheralDevices()	144
5.14.3 Member Function Documentation	144
5.14.3.1 clear_counter()	145
5.14.3.2 read()	145
5.14.3.3 read_counter()	146
5.14.3.4 read_fire_button_press()	147
5.14.3.5 start_motor()	148
5.14.3.6 stop_motor()	148
5.14.3.7 toggle_motor()	149
5.14.4 Member Data Documentation	150
5.14.4.1 CsRegister	150
5.14.4.2 motor_running	150
5.14.4.3 StatusReg	150
5.15 Screen_ctrl Class Reference	151
5.15.1 Detailed Description	152
5.15.2 Constructor & Destructor Documentation	152
5.15.2.1 Screen_ctrl()	152
5.15.2.2 ~Screen_ctrl()	152
5.15.3 Member Function Documentation	152
5.15.3.1 clean_screen()	153
5.15.3.2 get_time()	153
5.15.3.3 pass_to_ard()	153
5.15.3.4 print_logo()	154
5.15.3.5 print_version()	154
5.15.3.6 qa_mon()	155
5.15.3.7 tx_send_general_error()	155
5.15.3.8 update_battery()	156
5.15.3.9 update_error_remaining()	157
5.15.3.10 update_piazo()	158
5.15.3.11 update_pressure()	159
5.15.3.12 update_pulse_counter()	160
5.15.3.13 update_remaining_in_percent()	161
5.16 State Class Reference	162
5.16.1 Detailed Description	163
5.16.2 Constructor & Destructor Documentation	163
5.16.2.1 State()	163
5.16.2.2 ~State()	163
5.16.3 Member Function Documentation	163

5.16.3.1 init_first_step()	163
5.16.4 Member Data Documentation	164
5.16.4.1 battery	164
5.16.4.2 bypass	164
5.16.4.3 current	164
5.16.4.4 next	165
5.16.4.5 pressure	165
5.16.4.6 pulses_to_screen	165
5.17 STATUS_reg Class Reference	165
5.17.1 Detailed Description	166
5.17.2 Constructor & Destructor Documentation	167
5.17.2.1 STATUS_reg()	167
5.17.2.2 ~STATUS_reg()	167
5.17.3 Member Function Documentation	167
5.17.3.1 get_value()	167
5.17.3.2 print_current()	168
5.17.4 Member Data Documentation	168
5.17.4.1 current	168
6 File Documentation	169
6.1 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc.h File Reference	169
6.1.1 Function Documentation	170
6.1.1.1 MX_ADC_Init()	170
6.1.2 Variable Documentation	172
6.1.2.1 hadc	172
6.2 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc_controller.h File Reference	173
6.2.1 Enumeration Type Documentation	174
6.2.1.1 select_channel	174
6.3 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/common.h File Reference	174
6.3.1 Detailed Description	176
6.3.2 Macro Definition Documentation	177
6.3.2.1 CONVERTDEC	177
6.3.2.2 CONVERTHEX	177
6.3.2.3 CONVERTHEX_ALPHA	177
6.3.2.4 IS_09	177
6.3.2.5 IS_CAP_LETTER	177
6.3.2.6 IS_LC_LETTER	178
6.3.2.7 ISVALIDDEC	178
6.3.2.8 ISVALIDHEX	178
6.3.2.9 RX_TIMEOUT	178
6.3.2.10 TX_TIMEOUT	178
6.4 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/crc.h File Reference	179

6.4.1 Function Documentation	180
6.4.1.1 MX_CRC_Init()	180
6.4.2 Variable Documentation	180
6.4.2.1 hcrc	181
6.5 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h File Reference	181
6.5.1 Enumeration Type Documentation	182
6.5.1.1 cs_decoder	182
6.6 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/dma.h File Reference	183
6.6.1 Function Documentation	183
6.6.1.1 MX_DMA_Init()	184
6.7 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/flash_if.h File Reference	185
6.7.1 Detailed Description	186
6.7.2 Macro Definition Documentation	187
6.7.2.1 ABS_RETURN	187
6.7.2.2 APPLICATION_ADDRESS	187
6.7.2.3 FLASH_PAGE_STEP	187
6.7.2.4 USER_FLASH_SIZE	187
6.7.3 Enumeration Type Documentation	187
6.7.3.1 anonymous enum	187
6.7.3.2 anonymous enum	188
6.7.3.3 anonymous enum	188
6.8 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/gpio.h File Reference	189
6.8.1 Function Documentation	189
6.8.1.1 MX_GPIO_Init()	190
6.9 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/hardware.h File Reference	192
6.10 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main.h File Reference	194
6.10.1 Detailed Description	196
6.10.2 Macro Definition Documentation	196
6.10.2.1 ADC_CURRENT_SENSE_GPIO_Port	196
6.10.2.2 ADC_CURRENT_SENSE_Pin	196
6.10.2.3 BLINKING_LED_GPIO_Port	197
6.10.2.4 BLINKING_LED_Pin	197
6.10.2.5 BUZZER_GPIO_Port	197
6.10.2.6 BUZZER_Pin	197
6.10.2.7 CHIP_SELECT MCU_MEM_GPIO_Port	197
6.10.2.8 CHIP_SELECT MCU_MEM_Pin	197
6.10.2.9 CHIP_SELECT_RESERVED_GPIO_Port	198
6.10.2.10 CHIP_SELECT_RESERVED_Pin	198
6.10.2.11 CHIP_SELECT_SERIAL_GPIO_Port	198
6.10.2.12 CHIP_SELECT_SERIAL_Pin	198
6.10.2.13 DEBUG_AM_OK_GPIO_Port	198
6.10.2.14 DEBUG_AM_OK_Pin	198

6.10.2.15 DEBUG_APT_OK_GPIO_Port	199
6.10.2.16 DEBUG_APT_OK_Pin	199
6.10.2.17 DEBUG_BRK1_GPIO_Port	199
6.10.2.18 DEBUG_BRK1_Pin	199
6.10.2.19 DEBUG_BRK2_GPIO_Port	199
6.10.2.20 DEBUG_BRK2_Pin	199
6.10.2.21 DEBUG_CLK_AUX_GPIO_Port	200
6.10.2.22 DEBUG_CLK_AUX_Pin	200
6.10.2.23 DEBUG_DATA_AUX_GPIO_Port	200
6.10.2.24 DEBUG_DATA_AUX_Pin	200
6.10.2.25 DEBUG_LATCH_AUX_GPIO_Port	200
6.10.2.26 DEBUG_LATCH_AUX_Pin	200
6.10.2.27 DEBUG MCU_OK_GPIO_Port	201
6.10.2.28 DEBUG MCU_OK_Pin	201
6.10.2.29 DEBUG_MOTOR_RUNNING_GPIO_Port	201
6.10.2.30 DEBUG_MOTOR_RUNNING_Pin	201
6.10.2.31 DEBUG_PRESSURE_OK_GPIO_Port	201
6.10.2.32 DEBUG_PRESSURE_OK_Pin	201
6.10.2.33 DEBUG_STATE_1_GPIO_Port	202
6.10.2.34 DEBUG_STATE_1_Pin	202
6.10.2.35 DEBUG_STATE_2_GPIO_Port	202
6.10.2.36 DEBUG_STATE_2_Pin	202
6.10.2.37 DEBUG_STATE_3_GPIO_Port	202
6.10.2.38 DEBUG_STATE_3_Pin	202
6.10.2.39 DMA_LENGTH	203
6.10.2.40 ENABLE_POWER_12V_GPIO_Port	203
6.10.2.41 ENABLE_POWER_12V_Pin	203
6.10.2.42 LOGIC_BYPASS_1_GPIO_Port	203
6.10.2.43 LOGIC_BYPASS_1_Pin	203
6.10.2.44 LOGIC_BYPASS_2_GPIO_Port	203
6.10.2.45 LOGIC_BYPASS_2_Pin	204
6.10.2.46 LOGIC_BYPASS_3_GPIO_Port	204
6.10.2.47 LOGIC_BYPASS_3_Pin	204
6.10.2.48 LOGIC_BYPASS_4_GPIO_Port	204
6.10.2.49 LOGIC_BYPASS_4_Pin	204
6.10.2.50 LVDS_RESERVED_GPIO_Port	204
6.10.2.51 LVDS_RESERVED_Pin	205
6.10.2.52 PA1_RESERVED_ASK_SHAUL_GPIO_Port	205
6.10.2.53 PA1_RESERVED_ASK_SHAUL_Pin	205
6.10.2.54 POWER_FAIL MCU_GPIO_Port	205
6.10.2.55 POWER_FAIL MCU_Pin	205
6.10.2.56 RESERVED_UART6_RX_GPIO_Port	205

6.10.2.57 RESERVED_UART6_RX_Pin	206
6.10.2.58 RESERVED_UART6_TX_GPIO_Port	206
6.10.2.59 RESERVED_UART6_TX_Pin	206
6.10.2.60 RTC_TIMESTAMP_GPIO_Port	206
6.10.2.61 RTC_TIMESTAMP_Pin	206
6.10.2.62 SIMPLE_INTERLOCK_GPIO_Port	206
6.10.2.63 SIMPLE_INTERLOCK_Pin	207
6.10.2.64 SYS_SWO_RESERVED_GPIO_Port	207
6.10.2.65 SYS_SWO_RESERVED_Pin	207
6.10.2.66 TOGGLE400_RESET_BTN_GPIO_Port	207
6.10.2.67 TOGGLE400_RESET_BTN_Pin	207
6.10.2.68 UART1_PC_RX_GPIO_Port	207
6.10.2.69 UART1_PC_RX_Pin	208
6.10.2.70 UART1_PC_TX_GPIO_Port	208
6.10.2.71 UART1_PC_TX_Pin	208
6.10.2.72 UART2_SCREEN_RX_GPIO_Port	208
6.10.2.73 UART2_SCREEN_RX_Pin	208
6.10.2.74 UART2_SCREEN_TX_GPIO_Port	208
6.10.2.75 UART2_SCREEN_TX_Pin	209
6.10.2.76 USB_DM_RESERVED_GPIO_Port	209
6.10.2.77 USB_DM_RESERVED_Pin	209
6.10.2.78 USB_DP_RESERVED_GPIO_Port	209
6.10.2.79 USB_DP_RESERVED_Pin	209
6.10.2.80 WATCHDOG_OUT_GPIO_Port	209
6.10.2.81 WATCHDOG_OUT_Pin	210
6.10.3 Function Documentation	210
6.10.3.1 Error_Handler()	210
6.11 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main_controller.h File Reference	211
6.12 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem.h File Reference	212
6.13 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem_controller.h File Reference	213
6.14 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/menu.h File Reference	214
6.14.1 Detailed Description	216
6.14.2 Typedef Documentation	217
6.14.2.1 pFunction	217
6.14.3 Function Documentation	217
6.14.3.1 SerialDownload()	217
6.14.4 Variable Documentation	217
6.14.4.1 aFileName	217
6.15 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.cpp File Reference	217
6.16 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.h File Reference	218
6.17 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/peripheral_controller.h File Reference	218
6.18 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/rtc.h File Reference	219

6.18.1 Function Documentation	220
6.18.1.1 MX_RTC_Init()	221
6.18.2 Variable Documentation	222
6.18.2.1 hrtc	222
6.19 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/screen_controller.h File Reference	223
6.20 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/spi.h File Reference	224
6.20.1 Function Documentation	224
6.20.1.1 MX_SPI1_Init()	225
6.20.2 Variable Documentation	225
6.20.2.1 hspi1	225
6.21 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/state.h File Reference	226
6.21.1 Enumeration Type Documentation	227
6.21.1.1 statesMachine	227
6.22 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/status_reg.h File Reference	228
6.23 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_hal_conf.h File Reference	229
6.23.1 Detailed Description	231
6.23.2 Macro Definition Documentation	231
6.23.2.1 assert_param	231
6.23.2.2 DATA_CACHE_ENABLE	232
6.23.2.3 HAL_ADC_MODULE_ENABLED	232
6.23.2.4 HAL_CORTEX_MODULE_ENABLED	232
6.23.2.5 HAL_CRC_MODULE_ENABLED	232
6.23.2.6 HAL_DMA_MODULE_ENABLED	232
6.23.2.7 HAL_FLASH_MODULE_ENABLED	232
6.23.2.8 HAL_GPIO_MODULE_ENABLED	233
6.23.2.9 HAL_I2C_MODULE_ENABLED	233
6.23.2.10 HAL_MODULE_ENABLED	233
6.23.2.11 HAL_PWR_MODULE_ENABLED	233
6.23.2.12 HAL_RCC_MODULE_ENABLED	233
6.23.2.13 HAL_RTC_MODULE_ENABLED	233
6.23.2.14 HAL_SPI_MODULE_ENABLED	234
6.23.2.15 HAL_TIM_MODULE_ENABLED	234
6.23.2.16 HAL_UART_MODULE_ENABLED	234
6.23.2.17 HSE_STARTUP_TIMEOUT	234
6.23.2.18 HSE_VALUE	234
6.23.2.19 HSI14_VALUE	235
6.23.2.20 HSI48_VALUE	235
6.23.2.21 HSI_STARTUP_TIMEOUT	235
6.23.2.22 HSI_VALUE	235
6.23.2.23 INSTRUCTION_CACHE_ENABLE	236
6.23.2.24 LSE_STARTUP_TIMEOUT	236
6.23.2.25 LSE_VALUE	236

6.23.2.26 LSI_VALUE	236
6.23.2.27 PREFETCH_ENABLE	236
6.23.2.28 TICK_INT_PRIORITY	237
6.23.2.29 USERTOS	237
6.23.2.30 USE_SPI_CRC	237
6.23.2.31 VDD_VALUE	237
6.24 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_it.h File Reference	238
6.24.1 Detailed Description	238
6.24.2 Function Documentation	239
6.24.2.1 DMA1_Channel1_IRQHandler()	239
6.24.2.2 HardFault_Handler()	239
6.24.2.3 NMI_Handler()	240
6.24.2.4 PendSV_Handler()	240
6.24.2.5 SVC_Handler()	240
6.24.2.6 SysTick_Handler()	241
6.24.2.7 TIM16_IRQHandler()	241
6.24.2.8 TIM1_BRK_UP_TRG_COM_IRQHandler()	242
6.24.2.9 USART1_IRQHandler()	242
6.25 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/tim.h File Reference	242
6.25.1 Function Documentation	243
6.25.1.1 MX_TIM16_Init()	243
6.25.2 Variable Documentation	244
6.25.2.1 htim16	244
6.26 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/usart.h File Reference	245
6.26.1 Function Documentation	246
6.26.1.1 MX_USART1_UART_Init()	246
6.26.1.2 MX_USART2_UART_Init()	247
6.26.2 Variable Documentation	247
6.26.2.1 huart1	247
6.26.2.2 huart2	248
6.27 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/xmodem.h File Reference	248
6.27.1 Detailed Description	250
6.27.2 Macro Definition Documentation	251
6.27.2.1 ABORT1	251
6.27.2.2 ABORT2	251
6.27.2.3 ACK	251
6.27.2.4 CA	251
6.27.2.5 CRC16	251
6.27.2.6 DOWNLOAD_TIMEOUT	252
6.27.2.7 EOT	252
6.27.2.8 FILE_NAME_LENGTH	252
6.27.2.9 FILE_SIZE_LENGTH	252

6.27.2.10 MAX_ERRORS	252
6.27.2.11 NAK	252
6.27.2.12 NAK_TIMEOUT	253
6.27.2.13 NEGATIVE_BYTE	253
6.27.2.14 PACKET_1K_SIZE	253
6.27.2.15 PACKET_CNUMBER_INDEX	253
6.27.2.16 PACKET_DATA_INDEX	253
6.27.2.17 PACKET_HEADER_SIZE	253
6.27.2.18 PACKET_NUMBER_INDEX	254
6.27.2.19 PACKET_OVERHEAD_SIZE	254
6.27.2.20 PACKET_SIZE	254
6.27.2.21 PACKET_START_INDEX	254
6.27.2.22 PACKET_TRAILER_SIZE	254
6.27.2.23 SOH	254
6.27.2.24 STX	255
6.27.2.25 X_ACK	255
6.27.2.26 X_C	255
6.27.2.27 X_CAN	255
6.27.2.28 X_EOT	255
6.27.2.29 X_MAX_ERRORS	256
6.27.2.30 X_NAK	256
6.27.2.31 X_PACKET_1024_SIZE	256
6.27.2.32 X_PACKET_128_SIZE	256
6.27.2.33 X_PACKET_CRC_SIZE	256
6.27.2.34 X_PACKET_DATA_INDEX	256
6.27.2.35 X_PACKET_NUMBER_COMPLEMENT_INDEX	257
6.27.2.36 X_PACKET_NUMBER_INDEX	257
6.27.2.37 X_SOH	257
6.27.2.38 X_STX	257
6.27.3 Enumeration Type Documentation	257
6.27.3.1 xmodem_status	257
6.27.4 Function Documentation	258
6.27.4.1 xmodem_calc_crc()	258
6.27.4.2 xmodem_receive()	258
6.28 C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc.c File Reference	260
6.28.1 Function Documentation	261
6.28.1.1 HAL_ADC_MspDelInit()	261
6.28.1.2 HAL_ADC_MspInit()	262
6.28.1.3 MX_ADC_Init()	263
6.28.2 Variable Documentation	264
6.28.2.1 hadc	264
6.28.2.2 hdma_adc	265

6.29 C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc_controller.cpp File Reference	265
6.30 C:/Users/nicko/source/repos/APTX1_1/Core/Src/common.cpp File Reference	266
6.30.1 Detailed Description	266
6.31 C:/Users/nicko/source/repos/APTX1_1/Core/Src/crc.c File Reference	268
6.31.1 Function Documentation	268
6.31.1.1 HAL_CRC_MspDeInit()	269
6.31.1.2 HAL_CRC_MspInit()	269
6.31.1.3 MX_CRC_Init()	269
6.31.2 Variable Documentation	270
6.31.2.1 hcrc	270
6.32 C:/Users/nicko/source/repos/APTX1_1/Core/Src/cs_register.cpp File Reference	271
6.33 C:/Users/nicko/source/repos/APTX1_1/Core/Src/debug_facilities.cpp File Reference	271
6.34 C:/Users/nicko/source/repos/APTX1_1/Core/Src/dma.c File Reference	272
6.34.1 Function Documentation	273
6.34.1.1 MX_DMA_Init()	273
6.35 C:/Users/nicko/source/repos/APTX1_1/Core/Src/flash_if.cpp File Reference	274
6.35.1 Detailed Description	274
6.36 C:/Users/nicko/source/repos/APTX1_1/Core/Src/gpio.c File Reference	276
6.36.1 Function Documentation	276
6.36.1.1 MX_GPIO_Init()	276
6.37 C:/Users/nicko/source/repos/APTX1_1/Core/Src/main.cpp File Reference	278
6.37.1 Detailed Description	280
6.37.2 Macro Definition Documentation	280
6.37.2.1 __HAL_SYSCFG_REMAPMEMORY_SRAM	280
6.37.3 Function Documentation	280
6.37.3.1 __attribute__().	280
6.37.3.2 Error_Handler()	280
6.37.3.3 HAL_ADC_ConvCpltCallback()	281
6.37.3.4 HAL_TIM_PeriodElapsedCallback()	282
6.37.3.5 main()	282
6.37.3.6 remapMemToSRAM()	285
6.37.3.7 SystemClock_Config()	286
6.37.4 Variable Documentation	287
6.37.4.1 adc_controller	287
6.37.4.2 aPacketData	287
6.37.4.3 atp_extern	288
6.37.4.4 cs_register	288
6.37.4.5 cycle	288
6.37.4.6 debug	288
6.37.4.7 grand_ctrl	288
6.37.4.8 memory_am	288
6.37.4.9 memory_apt	289

6.37.4.10 memory_controller	289
6.37.4.11 memory_mcu	289
6.37.4.12 ph_device	289
6.37.4.13 reset_pressed	289
6.37.4.14 screen	289
6.37.4.15 state	290
6.37.4.16 status_register	290
6.37.4.17 timer_last_update	290
6.38 C:/Users/nicko/source/repos/APTX1_1/Core/Src/main_controller.cpp File Reference	290
6.38.1 Variable Documentation	291
6.38.1.1 reset_pressed	291
6.39 C:/Users/nicko/source/repos/APTX1_1/Core/Src/mem.cpp File Reference	291
6.40 C:/Users/nicko/source/repos/APTX1_1/Core/Src/mem_controller.cpp File Reference	291
6.41 C:/Users/nicko/source/repos/APTX1_1/Core/Src/menu.cpp File Reference	292
6.41.1 Detailed Description	293
6.42 C:/Users/nicko/source/repos/APTX1_1/Core/Src/peripheral_controller.cpp File Reference	294
6.43 C:/Users/nicko/source/repos/APTX1_1/Core/Src/rtc.c File Reference	295
6.43.1 Function Documentation	295
6.43.1.1 HAL_RTC_MspDeInit()	296
6.43.1.2 HAL_RTC_MspInit()	296
6.43.1.3 MX_RTC_Init()	296
6.43.2 Variable Documentation	297
6.43.2.1 hrtc	298
6.44 C:/Users/nicko/source/repos/APTX1_1/Core/Src/screen_controller.cpp File Reference	298
6.45 C:/Users/nicko/source/repos/APTX1_1/Core/Src/spi.c File Reference	299
6.45.1 Function Documentation	299
6.45.1.1 HAL_SPI_MspDeInit()	300
6.45.1.2 HAL_SPI_MspInit()	300
6.45.1.3 MX_SPI1_Init()	301
6.45.2 Variable Documentation	301
6.45.2.1 hspi1	301
6.46 C:/Users/nicko/source/repos/APTX1_1/Core/Src/state.cpp File Reference	302
6.47 C:/Users/nicko/source/repos/APTX1_1/Core/Src/status_reg.cpp File Reference	302
6.48 C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_hal_msp.c File Reference	303
6.48.1 Function Documentation	304
6.48.1.1 HAL_MspInit()	304
6.49 C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_hal_timebase_tim.c File Reference	304
6.49.1 Detailed Description	305
6.49.2 Function Documentation	306
6.49.2.1 HAL_InitTick()	306
6.49.2.2 HAL_ResumeTick()	307
6.49.2.3 HAL_SuspendTick()	307

6.49.3 Variable Documentation	308
6.49.3.1 htim1	308
6.50 C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_it.c File Reference	308
6.50.1 Detailed Description	309
6.50.2 Function Documentation	310
6.50.2.1 DMA1_Channel1_IRQHandler()	310
6.50.2.2 HardFault_Handler()	310
6.50.2.3 NMI_Handler()	310
6.50.2.4 PendSV_Handler()	311
6.50.2.5 read_reset_button()	311
6.50.2.6 SVC_Handler()	311
6.50.2.7 SysTick_Handler()	312
6.50.2.8 test_for_reset_press()	312
6.50.2.9 TIM16_IRQHandler()	313
6.50.2.10 TIM1_BRK_UP_TRG_COM_IRQHandler()	313
6.50.2.11 USART1_IRQHandler()	313
6.50.3 Variable Documentation	314
6.50.3.1 hdma_adc	314
6.50.3.2 htim1	314
6.50.3.3 htim16	314
6.50.3.4 huart1	314
6.50.3.5 reset_pressed	315
6.51 C:/Users/nicko/source/repos/APTX1_1/Core/Src/system_stm32f0xx.c File Reference	315
6.51.1 Detailed Description	316
6.51.2 3. This file configures the system clock as follows:	316
6.51.2.1 Supported STM32F0xx device	316
6.51.2.2 System Clock source HSI	316
6.51.2.3 SYSCLK(Hz) 8000000	316
6.51.2.4 HCLK(Hz) 8000000	316
6.51.2.5 AHB Prescaler 1	316
6.51.2.6 APB1 Prescaler 1	316
6.52 C:/Users/nicko/source/repos/APTX1_1/Core/Src/tim.c File Reference	317
6.52.1 Function Documentation	318
6.52.1.1 HAL_TIM_Base_MspDeInit()	318
6.52.1.2 HAL_TIM_Base_MspInit()	318
6.52.1.3 MX_TIM16_Init()	319
6.52.2 Variable Documentation	319
6.52.2.1 htim16	320
6.53 C:/Users/nicko/source/repos/APTX1_1/Core/Src/usart.c File Reference	320
6.53.1 Function Documentation	321
6.53.1.1 HAL_UART_MspDeInit()	321
6.53.1.2 HAL_UART_MspInit()	322

6.53.1.3 MX_USART1_UART_Init()	322
6.53.1.4 MX_USART2_UART_Init()	323
6.53.2 Variable Documentation	324
6.53.2.1 huart1	324
6.53.2.2 huart2	325
6.54 C:/Users/nicko/source/repos/APTX1_1/Core/Src/xmodem.cpp File Reference	325
6.54.1 Detailed Description	326
6.54.2 Function Documentation	326
6.54.2.1 xmodem_calc_crc()	326
6.54.2.2 xmodem_receive()	327
6.54.3 Variable Documentation	328
6.54.3.1 aPacketData	328
Index	329

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

CORE	7
Armenta	8
Globals	9
SPI	14
Auxilary	18
ADC	23
STM32L0xx_IAP_Main	27
STM32L0xx_IAP	31
CMSIS	36
Stm32f0xx_system	37
STM32F0xx_System_Private_Includes	38
STM32F0xx_System_Private_TypesDefinitions	39
STM32F0xx_System_Private_Defines	40
STM32F0xx_System_Private_Macros	41
STM32F0xx_System_Private_Variables	42
STM32F0xx_System_Private_FunctionPrototypes	43
STM32F0xx_System_Private_Functions	44

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

adc_command	47
AdcController	51
bypass_state	57
byte8_t_mem_status	59
byte8_t_reg_cs	64
byte8_t_reg_readback	67
byte8_t_reg_status	69
channel	72
CS_reg	74
MainController	80
Mem_ctrl	97
Memory	111
Monitor	122
PeripheralDevices	143
Screen_ctrl	151
State	162
STATUS_reg	165

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc.h	169
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc_controller.h	173
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/common.h This file provides all the headers of the common functions	174
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/crc.h	179
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h	181
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/dma.h	183
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/flash_if.h This file provides all the headers of the flash_if functions	185
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/gpio.h	189
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/hardware.h	192
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main.h : Header for main.c file. This file contains the common defines of the application	194
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main_controller.h	211
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem.h	212
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem_controller.h	213
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/menu.h This file provides all the headers of the menu functions	214
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.cpp	217
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.h	218
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/peripheral_controller.h	218
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/rtc.h	219
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/screen_controller.h	223
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/spi.h	224
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/state.h	226
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/status_reg.h	228
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_hal_conf.h HAL configuration file	229
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_it.h This file contains the headers of the interrupt handlers	238
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/tim.h	242
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/usart.h	245
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/xmodem.h This module is the implementation of the Xmodem protocol	248
C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc.c	260

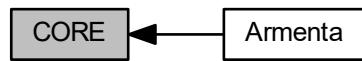
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ adc_controller.cpp	265
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ common.cpp	
This file provides all the common functions	266
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ crc.c	268
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ cs_register.cpp	271
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ debug_facilities.cpp	271
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ dma.c	272
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ flash_if.cpp	
This file provides all the memory related operation functions	274
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ gpio.c	276
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ main.cpp	
: Main program body	278
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ main_controller.cpp	290
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ mem.cpp	291
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ mem_controller.cpp	291
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ menu.cpp	
This file provides the software which contains the main menu routine. The main menu gives the options of:	292
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ peripheral_controller.cpp	294
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ rtc.c	295
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ screen_controller.cpp	298
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ spi.c	299
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ state.cpp	302
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ status_reg.cpp	302
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ stm32f0xx_hal_msp.c	303
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ stm32f0xx_hal_timebase_tim.c	
HAL time base based on the hardware TIM	304
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ stm32f0xx_it.c	
Interrupt Service Routines	308
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ system_stm32f0xx.c	
CMSIS Cortex-M0 Device Peripheral Access Layer System Source File	315
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ tim.c	317
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ uart.c	320
C:/Users/nicko/source/repos/APTX1_1/Core/Src/ xmodem.cpp	
This module is the implementation of the Xmodem protocol	325

Chapter 4

Module Documentation

4.1 CORE

Collaboration diagram for CORE:



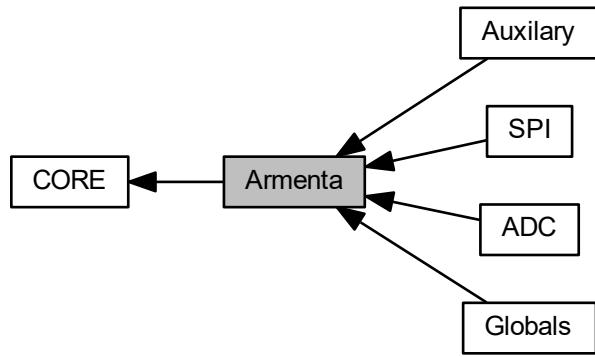
Modules

- [Armenta](#)

4.1.1 Detailed Description

4.2 Armenta

Collaboration diagram for Armenta:



Modules

- [Globals](#)
- [SPI](#)
- [Auxilary](#)
- [ADC](#)

4.2.1 Detailed Description

4.3 Globals

Collaboration diagram for Globals:



Macros

- #define DEBUG_PRINT 0
- #define TECHNICIAN_ENABLED 0
- #define PASS_SPI 0
- #define PASS_PRESSURE_CHECK 0
- #define EXPERIMENTAL 0
- #define DONT_BEEP 0
- #define FROM_BOOTLOADER 1
- #define MAX_HOLE_SIZE 5
- #define ADC_Presure 0
- #define ADC_Battery 1
- #define ADC_Presure_27_bar 500
- #define ADC_Presure_20_bar 400
- #define VERSION "7.3"
- #define MCU_VERSION_MAJOR 7
- #define MCU_VERSION_MINOR 3
- #define MCU_VERSION_PATCH 0
- #define MCU_VERSION_RC 1
- #define APP_ADDRESS (uint32_t)0x08008000
- #define APP_ADDRESS_P (uint32_t*)APP_ADDRESS
- #define END_ADDRESS (uint32_t)0x08020000
- #define APP_SIZE (uint32_t)(END_ADDRESS - APP_ADDRESS)

4.3.1 Detailed Description

4.3.2 Macro Definition Documentation

4.3.2.1 ADC_Battery

```
#define ADC_Battery 1
```

index of dma output to the array

Definition at line 24 of file hardware.h.

4.3.2.2 ADC_Presure

```
#define ADC_Presure 0
```

index of dma output to the array

Definition at line 23 of file hardware.h.

4.3.2.3 ADC_Presure_20_bar

```
#define ADC_Presure_20_bar 400
```

below this engine will stall while running

Definition at line 27 of file hardware.h.

4.3.2.4 ADC_Presure_27_bar

```
#define ADC_Presure_27_bar 500
```

below this engine will not start

Definition at line 26 of file hardware.h.

4.3.2.5 APP_ADDRESS

```
#define APP_ADDRESS (uint32_t)0x08008000
```

Definition at line 34 of file hardware.h.

4.3.2.6 APP_ADDRESS_P

```
#define APP_ADDRESS_P (uint32_t*)APP_ADDRESS
```

Definition at line 35 of file hardware.h.

4.3.2.7 APP_SIZE

```
#define APP_SIZE (uint32_t) (END_ADDRESS - APP_ADDRESS)
```

Definition at line 38 of file hardware.h.

4.3.2.8 DEBUG_PRINT

```
#define DEBUG_PRINT 0
```

TRUE - prints all globals, FALSE - no print out

Definition at line 14 of file hardware.h.

4.3.2.9 DONT_BEEP

```
#define DONT_BEEP 0
```

TRUE - disables beeper

Definition at line 19 of file hardware.h.

4.3.2.10 END_ADDRESS

```
#define END_ADDRESS (uint32_t) 0x08020000
```

End address of application space (address of last byte)

Definition at line 37 of file hardware.h.

4.3.2.11 EXPERIMENTAL

```
#define EXPERIMENTAL 0
```

TRUE - after 10k make an init done save

Definition at line 18 of file hardware.h.

4.3.2.12 FROM_BOOTLOADER

```
#define FROM_BOOTLOADER 1
```

TRUE - the application is launched from boot loader

Definition at line 20 of file hardware.h.

4.3.2.13 MAX_HOLE_SIZE

```
#define MAX_HOLE_SIZE 5
```

maximum size for missed writes consecutively

Definition at line 21 of file hardware.h.

4.3.2.14 MCU_VERSION_MAJOR

```
#define MCU_VERSION_MAJOR 7
```

Major version

Definition at line 29 of file hardware.h.

4.3.2.15 MCU_VERSION_MINOR

```
#define MCU_VERSION_MINOR 3
```

Minor version

Definition at line 30 of file hardware.h.

4.3.2.16 MCU_VERSION_PATCH

```
#define MCU_VERSION_PATCH 0
```

Patch version

Definition at line 31 of file hardware.h.

4.3.2.17 MCU_VERSION_RC

```
#define MCU_VERSION_RC 1
```

Release candidate version

Definition at line 32 of file hardware.h.

4.3.2.18 PASS_PRESSURE_CHECK

```
#define PASS_PRESSURE_CHECK 0
```

TRUE - works with low pressure

Definition at line 17 of file hardware.h.

4.3.2.19 PASS_SPI

```
#define PASS_SPI 0
```

TRUE - skips the spi init functionality

Definition at line 16 of file hardware.h.

4.3.2.20 TECHNICIAN_ENABLED

```
#define TECHNICIAN_ENABLED 0
```

TRUE - starts in UartDebug state

Definition at line 15 of file hardware.h.

4.3.2.21 VERSION

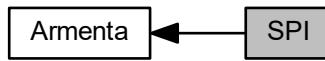
```
#define VERSION "7.3"
```

software version

Definition at line 28 of file hardware.h.

4.4 SPI

Collaboration diagram for SPI:



Macros

- #define READ_ARRAY1 0x0B
- #define READ_ARRAY2 0x03
- #define READ_DUAL 0x3B
- #define READ_QUAD 0x6B
- #define BLOCK_ERASE_4K 0x20
- #define BLOCK_ERASE_32K 0x52
- #define BLOCK_ERASE_64K 0xD8
- #define ERASE_CHIP1 0x60
- #define ERASE_CHIP2 0xC7
- #define BYTE_PROGRAM 0x02
- #define WRITE_ENABLE 0x06
- #define WRITE_DISABLE 0x04
- #define READ_STATUS_REGISTER1 0x05
- #define READ_STATUS_REGISTER2 0x35
- #define WRITE_STATUS_REGISTER 0x01
- #define WRITE_ENABLE_STATUS_REGISTER_VOLATILE 0x50
- #define READ_MANUFACTURE_ID 0x9F
- #define READ_DEVICE_ID 0x90
- #define HIGH (GPIO_PinState)1
- #define LOW (GPIO_PinState)0

4.4.1 Detailed Description

4.4.2 Macro Definition Documentation

4.4.2.1 BLOCK_ERASE_32K

```
#define BLOCK_ERASE_32K 0x52
```

Definition at line 51 of file hardware.h.

4.4.2.2 BLOCK_ERASE_4K

```
#define BLOCK_ERASE_4K 0x20
```

Definition at line 50 of file hardware.h.

4.4.2.3 BLOCK_ERASE_64K

```
#define BLOCK_ERASE_64K 0xD8
```

Definition at line 52 of file hardware.h.

4.4.2.4 BYTE_PROGRAM

```
#define BYTE_PROGRAM 0x02
```

Definition at line 56 of file hardware.h.

4.4.2.5 ERASE_CHIP1

```
#define ERASE_CHIP1 0x60
```

Definition at line 53 of file hardware.h.

4.4.2.6 ERASE_CHIP2

```
#define ERASE_CHIP2 0xC7
```

Definition at line 54 of file hardware.h.

4.4.2.7 HIGH

```
#define HIGH (GPIO_PinState)1
```

Definition at line 69 of file hardware.h.

4.4.2.8 LOW

```
#define LOW (GPIO_PinState)0
```

Definition at line 70 of file hardware.h.

4.4.2.9 READ_ARRAY1

```
#define READ_ARRAY1 0x0B
```

Definition at line 45 of file hardware.h.

4.4.2.10 READ_ARRAY2

```
#define READ_ARRAY2 0x03
```

Definition at line 46 of file hardware.h.

4.4.2.11 READ_DEVICE_ID

```
#define READ_DEVICE_ID 0x90
```

Definition at line 67 of file hardware.h.

4.4.2.12 READ_DUAL

```
#define READ_DUAL 0x3B
```

Definition at line 47 of file hardware.h.

4.4.2.13 READ_MANUFACTURE_ID

```
#define READ_MANUFACTURE_ID 0x9F
```

Definition at line 66 of file hardware.h.

4.4.2.14 READ_QUAD

```
#define READ_QUAD 0x6B
```

Definition at line 48 of file hardware.h.

4.4.2.15 READ_STATUS_REGISTER1

```
#define READ_STATUS_REGISTER1 0x05
```

Definition at line 61 of file hardware.h.

4.4.2.16 READ_STATUS_REGISTER2

```
#define READ_STATUS_REGISTER2 0x35
```

Definition at line 62 of file hardware.h.

4.4.2.17 WRITE_DISABLE

```
#define WRITE_DISABLE 0x04
```

Definition at line 59 of file hardware.h.

4.4.2.18 WRITE_ENABLE

```
#define WRITE_ENABLE 0x06
```

Definition at line 58 of file hardware.h.

4.4.2.19 WRITE_ENABLE_STATUS_REGISTER_VOLATILE

```
#define WRITE_ENABLE_STATUS_REGISTER_VOLATILE 0x50
```

Definition at line 64 of file hardware.h.

4.4.2.20 WRITE_STATUS_REGISTER

```
#define WRITE_STATUS_REGISTER 0x01
```

Definition at line 63 of file hardware.h.

4.5 Auxilary

Collaboration diagram for Auxilary:



Macros

- #define HDC_1080_ADD 0x40
- #define Configuration_register_add 0x02
- #define Temperature_register_add 0x00
- #define Humidity_register_add 0x01
- #define START_BEEP 100
- #define BEEP_COUNTER_200 110
- #define BEEP_COUNTER_400 120
- #define BEEP_COUNTER_0 130
- #define KEY_PRESS 80
- #define Beep_Every_X_Counting 200
- #define SERIAL_PC &huart1
- #define SERIAL_SCREEN &huart2
- #define RXBUFFERSIZE 256
- #define TIMEOUT_SCREEN 5000
- #define TICKS_FOR_UPDATE 500
- #define TICKS_SCREEN_UPDATE_LONG 1250
- #define TICKS_SCREEN_UPDATE_SUPER_LONG 3500
- #define AM_WARNING_TIMES 3
- #define MOTOR_1_PWM_ON_50 50
- #define MOTOR_1_PWM_ON_75 75
- #define MOTOR_1_PWM_ON 100
- #define MOTOR_1_PWM_OFF 0

4.5.1 Detailed Description

4.5.2 Macro Definition Documentation

4.5.2.1 AM_WARNING_TIMES

```
#define AM_WARNING_TIMES 3
```

Definition at line 98 of file hardware.h.

4.5.2.2 BEEP_COUNTER_0

```
#define BEEP_COUNTER_0 130
```

Definition at line 85 of file hardware.h.

4.5.2.3 BEEP_COUNTER_200

```
#define BEEP_COUNTER_200 110
```

Definition at line 83 of file hardware.h.

4.5.2.4 BEEP_COUNTER_400

```
#define BEEP_COUNTER_400 120
```

Definition at line 84 of file hardware.h.

4.5.2.5 Beep_Every_X_Counting

```
#define Beep_Every_X_Counting 200
```

Definition at line 88 of file hardware.h.

4.5.2.6 Configuration_register_add

```
#define Configuration_register_add 0x02
```

Definition at line 78 of file hardware.h.

4.5.2.7 HDC_1080_ADD

```
#define HDC_1080_ADD 0x40
```

Definition at line 77 of file hardware.h.

4.5.2.8 Humidity_register_add

```
#define Humidity_register_add 0x01
```

Definition at line 80 of file hardware.h.

4.5.2.9 KEY_PRESS

```
#define KEY_PRESS 80
```

Definition at line 86 of file hardware.h.

4.5.2.10 MOTOR_1_PWM_OFF

```
#define MOTOR_1_PWM_OFF 0
```

Definition at line 105 of file hardware.h.

4.5.2.11 MOTOR_1_PWM_ON

```
#define MOTOR_1_PWM_ON 100
```

Definition at line 104 of file hardware.h.

4.5.2.12 MOTOR_1_PWM_ON_50

```
#define MOTOR_1_PWM_ON_50 50
```

Definition at line 102 of file hardware.h.

4.5.2.13 MOTOR_1_PWM_ON_75

```
#define MOTOR_1_PWM_ON_75 75
```

Definition at line 103 of file hardware.h.

4.5.2.14 RXBUFFERSIZE

```
#define RXBUFFERSIZE 256
```

Definition at line 92 of file hardware.h.

4.5.2.15 SERIAL_PC

```
#define SERIAL_PC &huart1
```

Definition at line 90 of file hardware.h.

4.5.2.16 SERIAL_SCREEN

```
#define SERIAL_SCREEN &huart2
```

Definition at line 91 of file hardware.h.

4.5.2.17 START_BEEP

```
#define START_BEEP 100
```

Definition at line 82 of file hardware.h.

4.5.2.18 Temperature_register_add

```
#define Temperature_register_add 0x00
```

Definition at line 79 of file hardware.h.

4.5.2.19 TICKS_FOR_UPDATE

```
#define TICKS_FOR_UPDATE 500
```

Definition at line 95 of file hardware.h.

4.5.2.20 **TICKS_SCREEN_UPDATE_LONG**

```
#define TICKS_SCREEN_UPDATE_LONG 1250
```

Definition at line 96 of file hardware.h.

4.5.2.21 **TICKS_SCREEN_UPDATE_SUPER_LONG**

```
#define TICKS_SCREEN_UPDATE_SUPER_LONG 3500
```

Definition at line 97 of file hardware.h.

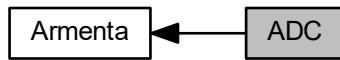
4.5.2.22 **TIMEOUT_SCREEN**

```
#define TIMEOUT_SCREEN 5000
```

Definition at line 94 of file hardware.h.

4.6 ADC

Collaboration diagram for ADC:



Macros

- #define SELECT_CHANNEL_0 0b0000
- #define SELECT_CHANNEL_1 0b0001
- #define SELECT_CHANNEL_2 0b0010
- #define SELECT_CHANNEL_3 0b0011
- #define SELECT_CHANNEL_4 0b0100
- #define SELECT_CHANNEL_5 0b0101
- #define SELECT_CHANNEL_6 0b0110
- #define SELECT_CHANNEL_7 0b0111
- #define ADC_RESERVED 0b1000
- #define ADC_RESERVED2 0b1001
- #define WRITE_CFR 0b1010
- #define SELECT_TEST_VDIV2 0b1011
- #define SELECT_TEST_REFM 0b1100
- #define SELECT_TEST_REFP 0b1101
- #define FIFOREAD 0b1110
- #define HARDWARE_DEFAULT 0b1111

4.6.1 Detailed Description

4.6.2 Macro Definition Documentation

4.6.2.1 ADC_RESERVED

```
#define ADC_RESERVED 0b1000
```

Definition at line 120 of file hardware.h.

4.6.2.2 ADC_RESERVED2

```
#define ADC_RESERVED2 0b1001
```

Definition at line 121 of file hardware.h.

4.6.2.3 FIFOREAD

```
#define FIFOREAD 0b1110
```

Definition at line 128 of file hardware.h.

4.6.2.4 HARDWARE_DEFAULT

```
#define HARDWARE_DEFAULT 0b1111
```

Definition at line 129 of file hardware.h.

4.6.2.5 SELECT_CHANNEL_0

```
#define SELECT_CHANNEL_0 0b0000
```

Definition at line 111 of file hardware.h.

4.6.2.6 SELECT_CHANNEL_1

```
#define SELECT_CHANNEL_1 0b0001
```

Definition at line 112 of file hardware.h.

4.6.2.7 SELECT_CHANNEL_2

```
#define SELECT_CHANNEL_2 0b0010
```

Definition at line 113 of file hardware.h.

4.6.2.8 SELECT_CHANNEL_3

```
#define SELECT_CHANNEL_3 0b0011
```

Definition at line 114 of file hardware.h.

4.6.2.9 SELECT_CHANNEL_4

```
#define SELECT_CHANNEL_4 0b0100
```

Definition at line 115 of file hardware.h.

4.6.2.10 SELECT_CHANNEL_5

```
#define SELECT_CHANNEL_5 0b0101
```

Definition at line 116 of file hardware.h.

4.6.2.11 SELECT_CHANNEL_6

```
#define SELECT_CHANNEL_6 0b0110
```

Definition at line 117 of file hardware.h.

4.6.2.12 SELECT_CHANNEL_7

```
#define SELECT_CHANNEL_7 0b0111
```

Definition at line 118 of file hardware.h.

4.6.2.13 SELECT_TEST_REFM

```
#define SELECT_TEST_REFM 0b1100
```

Definition at line 125 of file hardware.h.

4.6.2.14 **SELECT_TEST_REFP**

```
#define SELECT_TEST_REFP 0b1101
```

Definition at line 126 of file hardware.h.

4.6.2.15 **SELECT_TEST_VDIV2**

```
#define SELECT_TEST_VDIV2 0b1011
```

Definition at line 124 of file hardware.h.

4.6.2.16 **WRITE_CFR**

```
#define WRITE_CFR 0b1010
```

Definition at line 123 of file hardware.h.

4.7 STM32L0xx_IAP_Main

Functions

- void **Int2Str** (uint8_t *p_str, uint32_t intnum)
Convert an Integer to a string.
- uint32_t **Str2Int** (uint8_t *p_inputstr, uint32_t *p_intnum)
Convert a string to an integer.
- void **Serial_PutString** (const char *p_string)
Print a string on the HyperTerminal.
- HAL_StatusTypeDef **Serial_PutByte** (uint8_t param)
Transmit a byte to the HyperTerminal.

4.7.1 Detailed Description

4.7.2 Function Documentation

4.7.2.1 Int2Str()

```
void Int2Str (
    uint8_t * p_str,
    uint32_t intnum )
```

Convert an Integer to a string.

Parameters

<i>p_str</i>	The string output pointer
<i>intnum</i>	The integer to be converted

Return values

<i>None</i>	
-------------	--

Definition at line 61 of file common.cpp.

```
62 {
63     uint32_t i, divider = 1000000000, pos = 0, status = 0;
64
65     for (i = 0; i < 10; i++)
66     {
67         p_str[pos++] = (intnum / divider) + 48;
68
69         intnum = intnum % divider;
70         divider /= 10;
71         if ((p_str[pos-1] == '0') & (status == 0))
72         {
73             pos = 0;
74         }
75     else
76     {
77         status++;
78     }
79 }
```

```
79     }
80 }
```

4.7.2.2 Serial_PutByte()

```
HAL_StatusTypeDef Serial_PutByte (
    uint8_t param )
```

Transmit a byte to the HyperTerminal.

Parameters

<i>param</i>	The byte to be sent
--------------	---------------------

Return values

<i>HAL_StatusTypeDef</i>	HAL_OK if OK
--------------------------	--------------

Definition at line 180 of file common.cpp.

```
181 {
182
183     return HAL_UART_Transmit(SERIAL_PC, &param, 1, TX_TIMEOUT);
184 }
```

4.7.2.3 Serial_PutString()

```
void Serial_PutString (
    const char * p_string )
```

Print a string on the HyperTerminal.

Parameters

<i>p_string</i>	The string to be printed
-----------------	--------------------------

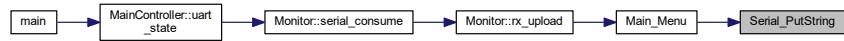
Return values

<i>None</i>	
-------------	--

Definition at line 164 of file common.cpp.

```
165 {
166     uint16_t length = 0;
167
168     while (p_string[length] != '\0')
169     {
170         length++;
171     }
172     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)p_string, length, TX_TIMEOUT);
173 }
```

Here is the caller graph for this function:



4.7.2.4 Str2Int()

```
uint32_t Str2Int (
    uint8_t * p_inputstr,
    uint32_t * p_intnum )
```

Convert a string to an integer.

Parameters

<i>p_inputstr</i>	The string to be converted
<i>p_intnum</i>	The integer value

Return values

1	Correct 0: Error
---	------------------

Definition at line 89 of file common.cpp.

```
90 {
91     uint32_t i = 0, res = 0;
92     uint32_t val = 0;
93
94     if ((p_inputstr[0] == '0') && ((p_inputstr[1] == 'x') || (p_inputstr[1] == 'X')))
95     {
96         i = 2;
97         while ((i < 11) && (p_inputstr[i] != '\0'))
98         {
99             if (ISVALIDHEX(p_inputstr[i]))
100             {
101                 val = (val << 4) + CONVERTHEX(p_inputstr[i]);
102             }
103             else
104             {
105                 /* Return 0, Invalid input */
106                 res = 0;
107                 break;
108             }
109             i++;
110         }
111
112         /* valid result */
113         if (p_inputstr[i] == '\0')
114         {
115             *p_intnum = val;
116             res = 1;
117         }
118     }
119     else /* max 10-digit decimal input */
120     {
121         while ((i < 11) && (res != 1))
122         {
123             if (p_inputstr[i] == '\0')
124             {
125                 *p_intnum = val;
126                 /* return 1 */
127             }
128         }
129     }
130 }
```

```
127         res = 1;
128     }
129     else if (((p_inputstr[i] == 'k') || (p_inputstr[i] == 'K')) && (i > 0))
130     {
131         val = val << 10;
132         *p_intnum = val;
133         res = 1;
134     }
135     else if (((p_inputstr[i] == 'm') || (p_inputstr[i] == 'M')) && (i > 0))
136     {
137         val = val << 20;
138         *p_intnum = val;
139         res = 1;
140     }
141     else if (ISVALIDDEC(p_inputstr[i]))
142     {
143         val = val * 10 + CONVERTDEC(p_inputstr[i]);
144     }
145     else
146     {
147         /* return 0, Invalid input */
148         res = 0;
149         break;
150     }
151     i++;
152 }
153 }
154 return res;
155 }
```

4.8 STM32L0xx_IAP

Functions

- void **FLASH_If_Erase** (void)
This function does an erase of all user flash area.
 - uint32_t **FLASH_If_Write** (uint32_t destination, uint8_t *p_source, uint32_t length)
This function writes a data buffer in flash (data are 32-bit aligned).
 - void **dump_mem** ()
 - void **Main_Menu** (void)
- Display the Main Menu on HyperTerminal.*

Variables

- Mem_ctrl * memory_extern
- pFunction JumpToApplication
- uint32_t JumpAddress
- uint32_t FlashProtection = 0
- uint8_t aFileName [FILE_NAME_LENGTH]
- Mem_ctrl * memory_extern

4.8.1 Detailed Description

4.8.2 Function Documentation

4.8.2.1 dump_mem()

```
void dump_mem ( )
```

Definition at line 74 of file menu.cpp.

```
75 {
76     uint8_t dump_c[40];
77     for (int i = 0; i < APP_SIZE; i++)
78     {
79         if (i % 16 == 0 && i > 0)
80         {
81             sprintf((char*)dump_c, " 0x%02x \r\n", *(volatile uint8_t*)(APP_ADDRESS+i));
82         }
83         else
84         {
85             sprintf((char*)dump_c, " 0x%02x", *(volatile uint8_t*)(APP_ADDRESS + i));
86         }
87         HAL_UART_Transmit(SERIAL_PC, dump_c, strlen((char*)dump_c), 10);
88     }
89 }
```

Here is the caller graph for this function:



4.8.2.2 FLASH_If_Erase()

```
void FLASH_If_Erase (
    void )
```

This function does an erase of all user flash area.

Parameters

<i>start</i>	start of user flash area
--------------	--------------------------

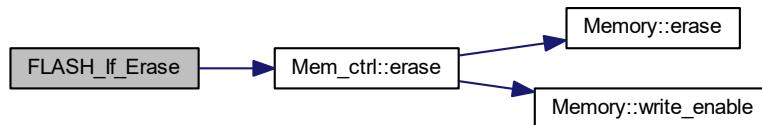
Return values

<i>FLASHIF_OK</i>	: user flash area successfully erased	<i>FLASHIF_ERASEKO</i> : error occurred
-------------------	---------------------------------------	---

Definition at line 62 of file flash_if.cpp.

```
63 {
64     memory_extern->erase(1);
65 }
```

Here is the call graph for this function:



4.8.2.3 FLASH_If_Write()

```
uint32_t FLASH_If_Write (
    uint32_t destination,
    uint8_t * p_source,
    uint32_t length )
```

This function writes a data buffer in flash (data are 32-bit aligned).

Note

After writing data buffer, the flash content is checked.

Parameters

<i>destination</i>	start address for target location
<i>p_source</i>	pointer on buffer with data to write
<i>length</i>	length of data buffer (unit is 32-bit word)

Return values

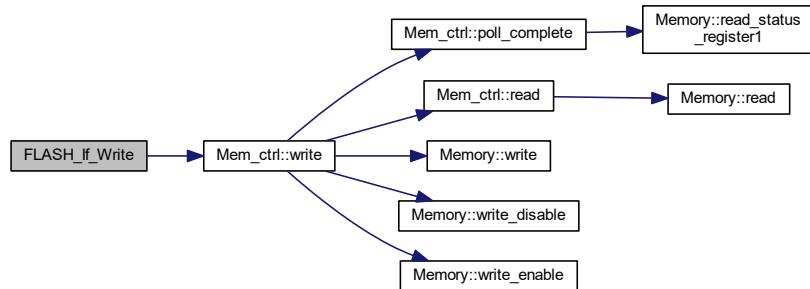
<code>uint32_t</code>	0: Data successfully written to Flash memory 1: Error occurred while writing data in Flash memory 2: Written Data in flash memory is different from expected one
-----------------------	--

Definition at line 78 of file flash_if.cpp.

```

79 {
80     for (uint32_t i = 0; i < length; i++)
81     {
82         uint32_t value = p_source[i*4]«24 | p_source[i*4+1]«16 | p_source[i*4+2]«8 | p_source[i*4+3];
83         memory_extern->write(1, destination + i * 4, value);
84     }
85     HAL_Delay(30);
86     return HAL_OK;
87 }
```

Here is the call graph for this function:



4.8.2.4 Main_Menu()

```
void Main_Menu (
    void )
```

Display the Main Menu on HyperTerminal.

Parameters

<code>None</code>	<input type="checkbox"/>
-------------------	--------------------------

Return values

<code>None</code>	<input type="checkbox"/>
-------------------	--------------------------

Definition at line 96 of file menu.cpp.

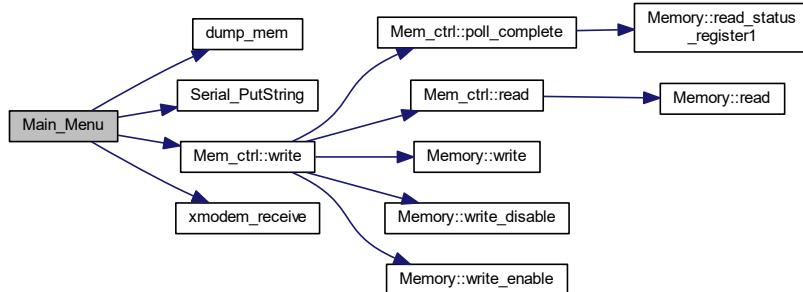
```

97 {
98     uint8_t key = 0;
99
100    Serial_PutString("\r\n=====");                                (C) COPYRIGHT 2019 ARMenta
101    Serial_PutString("\r\n=");
```

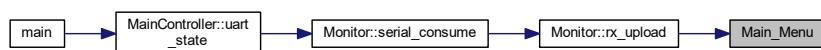
```

102     Serial_PutString("\r\n=");
103     Serial_PutString("\r\n= STM32F0xx In-Application Programming Application (Version 1.0.0) =");
104     Serial_PutString("\r\n= =====");
105     Serial_PutString("\r\n\r\n");
106
107
108
109 while (1)
110 {
111     Serial_PutString("\r\n===== Main Menu =====\r\n\r\n");
112     Serial_PutString(" Download image to the internal Flash ----- 1\r\n\r\n");
113     Serial_PutString(" Download the image of the internal Flash ----- 2\r\n\r\n");
114     Serial_PutString(" Execute the loaded application ----- 3\r\n\r\n");
115     Serial_PutString("===== \r\n\r\n");
116
117     /* Clean the input path */
118     //__HAL_UART_FLUSH_DRREGISTER(SERIAL_PC);
119     //__HAL_UART_CLEAR_IT(SERIAL_PC, UART_CLEAR_OREF);
120
121     /* Receive key */
122     HAL_UART_Receive(SERIAL_PC, &key, 1, RX_TIMEOUT);
123
124     switch (key)
125     {
126     case '1':
127         /* Download user application in the Flash */
128         xmodem_receive();
129         break;
130     case '2':
131         dump_mem();
132         break;
133     case '3':
134         Serial_PutString("Start program execution.....\r\n\r\n");
135         memory_extern->write(1, 0xffff00, 0x0f0f0f0f);
136         NVIC_SystemReset();
137         break;
138     default:
139         Serial_PutString("Invalid Number ! ==> The number should be either 1 or 2\r\n");
140         break;
141     }
142 }
143 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.8.3 Variable Documentation

4.8.3.1 aFileName

```
uint8_t aFileName[FILE_NAME_LENGTH]
```

4.8.3.2 FlashProtection

```
uint32_t FlashProtection = 0
```

Definition at line 65 of file menu.cpp.

4.8.3.3 JumpAddress

```
uint32_t JumpAddress
```

Definition at line 64 of file menu.cpp.

4.8.3.4 JumpToApplication

```
pFunction JumpToApplication
```

Definition at line 63 of file menu.cpp.

4.8.3.5 memory_extern [1/2]

```
Mem_ctrl* memory_extern
```

Definition at line 82 of file main.cpp.

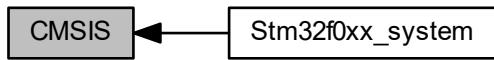
4.8.3.6 memory_extern [2/2]

```
Mem_ctrl* memory_extern
```

Definition at line 82 of file main.cpp.

4.9 CMSIS

Collaboration diagram for CMSIS:



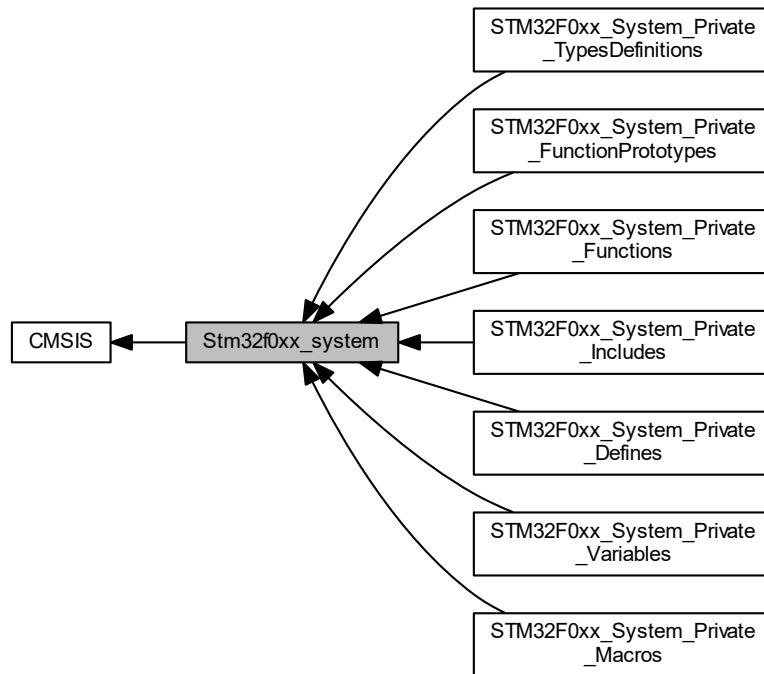
Modules

- [Stm32f0xx_system](#)

4.9.1 Detailed Description

4.10 Stm32f0xx_system

Collaboration diagram for Stm32f0xx_system:



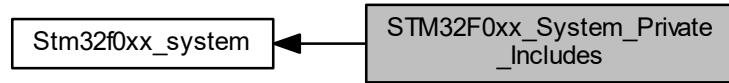
Modules

- [STM32F0xx_System_Private_Includes](#)
- [STM32F0xx_System_Private_TypesDefinitions](#)
- [STM32F0xx_System_Private_Defines](#)
- [STM32F0xx_System_Private_Macros](#)
- [STM32F0xx_System_Private_Variables](#)
- [STM32F0xx_System_Private_FunctionPrototypes](#)
- [STM32F0xx_System_Private_Functions](#)

4.10.1 Detailed Description

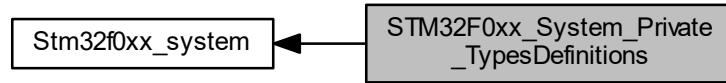
4.11 STM32F0xx_System_Private_Includes

Collaboration diagram for STM32F0xx_System_Private_Includes:



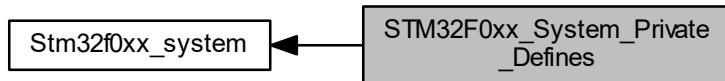
4.12 STM32F0xx_System_Private_TypesDefinitions

Collaboration diagram for STM32F0xx_System_Private_TypesDefinitions:



4.13 STM32F0xx_System_Private_Defines

Collaboration diagram for STM32F0xx_System_Private_Defines:



Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

4.13.1 Detailed Description

4.13.2 Macro Definition Documentation

4.13.2.1 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Default value of the External oscillator in Hz. This value can be provided and adapted by the user application.

Definition at line 100 of file system_stm32f0xx.c.

4.13.2.2 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000)
```

Default value of the HSI48 Internal oscillator in Hz. This value can be provided and adapted by the user application.

Definition at line 112 of file system_stm32f0xx.c.

4.13.2.3 HSI_VALUE

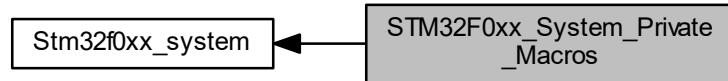
```
#define HSI_VALUE ((uint32_t)8000000)
```

Default value of the Internal oscillator in Hz. This value can be provided and adapted by the user application.

Definition at line 106 of file system_stm32f0xx.c.

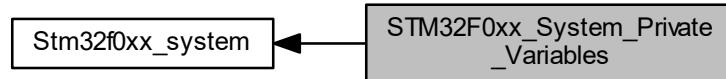
4.14 STM32F0xx_System_Private_Macros

Collaboration diagram for STM32F0xx_System_Private_Macros:



4.15 STM32F0xx_System_Private_Variables

Collaboration diagram for STM32F0xx_System_Private_Variables:



Variables

- `uint32_t SystemCoreClock = 8000000`
- `const uint8_t AHBPrescTable [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}`
- `const uint8_t APBPrescTable [8] = {0, 0, 0, 0, 1, 2, 3, 4}`

4.15.1 Detailed Description

4.15.2 Variable Documentation

4.15.2.1 AHBPrescTable

```
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
```

Definition at line 141 of file system_stm32f0xx.c.

4.15.2.2 APBPrescTable

```
const uint8_t APBPrescTable[8] = {0, 0, 0, 0, 1, 2, 3, 4}
```

Definition at line 142 of file system_stm32f0xx.c.

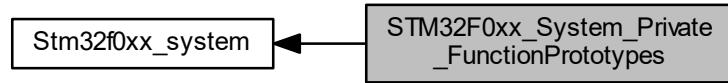
4.15.2.3 SystemCoreClock

```
uint32_t SystemCoreClock = 8000000
```

Definition at line 139 of file system_stm32f0xx.c.

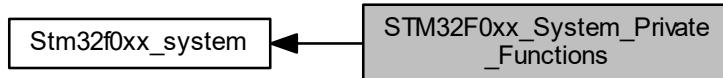
4.16 STM32F0xx_System_Private_FunctionPrototypes

Collaboration diagram for STM32F0xx_System_Private_FunctionPrototypes:



4.17 STM32F0xx_System_Private_Functions

Collaboration diagram for STM32F0xx_System_Private_Functions:



Functions

- void [SystemInit](#) (void)
Setup the microcontroller system. Initialize the default HSI clock source, vector table location and the PLL configuration is reset.
- void [SystemCoreClockUpdate](#) (void)
Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

4.17.1 Detailed Description

4.17.2 Function Documentation

4.17.2.1 SystemCoreClockUpdate()

```
void SystemCoreClockUpdate (
    void )
```

Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Note

Each time the core clock (HCLK) changes, this function must be called to update SystemCoreClock variable value. Otherwise, any configuration based on this variable will be incorrect.

- The system frequency computed by this function is not the real frequency in the chip. It is calculated based on the predefined constant and the selected clock source:

- If SYSCLK source is HSI, SystemCoreClock will contain the [HSI_VALUE\(*\)](#)
- If SYSCLK source is HSE, SystemCoreClock will contain the [HSE_VALUE\(**\)](#)
- If SYSCLK source is PLL, SystemCoreClock will contain the [HSE_VALUE\(**\)](#) or [HSI_VALUE\(*\)](#) multiplied/divided by the PLL factors.

(*) HSI_VALUE is a constant defined in `stm32f0xx_hal.h` file (default value 8 MHz) but the real value may vary depending on the variations in voltage and temperature.

(**) HSE_VALUE is a constant defined in `stm32f0xx_hal.h` file (default value 8 MHz), user has to ensure that HSE_VALUE is same as the real frequency of the crystal used. Otherwise, this function may have wrong result.

- The result of this function could be not correct when using fractional value for HSE crystal.

Parameters

None	<input type="button" value=""/>
------	---------------------------------

Return values

None	<input type="button" value=""/>
------	---------------------------------

Definition at line 263 of file system_stm32f0xx.c.

```

268     {
269     case RCC_CFGR_SWS_HSI: /* HSI used as system clock */
270         SystemCoreClock = HSI_VALUE;
271         break;
272     case RCC_CFGR_SWS_HSE: /* HSE used as system clock */
273         SystemCoreClock = HSE_VALUE;
274         break;
275     case RCC_CFGR_SWS_PLL: /* PLL used as system clock */
276         /* Get PLL clock source and multiplication factor -----*/
277         pllmull = RCC->CFGR & RCC_CFGR_PLLMUL;
278         pllsource = RCC->CFGR & RCC_CFGR_PLLSRC;
279         pllmull = ( pllmull » 18 ) + 2;
280         predivfactor = (RCC->CFGR2 & RCC_CFGR2_PREDIV) + 1;
281
282         if (pllsource == RCC_CFGR_PLLSRC_HSE_PREDIV)
283         {
284             /* HSE used as PLL clock source : SystemCoreClock = HSE/PREDIV * PLLMUL */
285             SystemCoreClock = (HSE_VALUE/predivfactor) * pllmull;
286         }
287 #if defined(STM32F042x6) || defined(STM32F048xx) || defined(STM32F072xB) || defined(STM32F078xx) ||
288     defined(STM32F091xC) || defined(STM32F098xx)
289     else if (pllsource == RCC_CFGR_PLLSRC_HSI48_PREDIV)
290     {
291         /* HSI48 used as PLL clock source : SystemCoreClock = HSI48/PREDIV * PLLMUL */
292         SystemCoreClock = (HSI48_VALUE/predivfactor) * pllmull;
293     }
294 #endif /* STM32F042x6 || STM32F048xx || STM32F072xB || STM32F078xx || STM32F091xC || STM32F098xx */
295     else
296     {
297 #if defined(STM32F042x6) || defined(STM32F048xx) || defined(STM32F070x6) \
298     || defined(STM32F078xx) || defined(STM32F071xB) || defined(STM32F072xB) \
299     || defined(STM32F070xB) || defined(STM32F091xC) || defined(STM32F098xx) || defined(STM32F030xC)
300         /* HSI used as PLL clock source : SystemCoreClock = HSI/PREDIV * PLLMUL */
301         SystemCoreClock = (HSI_VALUE/predivfactor) * pllmull;
302     else
303         /* HSI used as PLL clock source : SystemCoreClock = HSI/2 * PLLMUL */
304         SystemCoreClock = (HSI_VALUE » 1) * pllmull;
305     #endif /* STM32F042x6 || STM32F048xx || STM32F070x6 ||
306         STM32F071xB || STM32F072xB || STM32F078xx || STM32F070xB ||
307         STM32F091xC || STM32F098xx || STM32F030xC */
308     }
309     break;
310     default: /* HSI used as system clock */
311         SystemCoreClock = HSI_VALUE;
312         break;
313     /* Compute HCLK clock frequency -----*/
314     /* Get HCLK prescaler */
315     tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) » 4)];
316     /* HCLK clock frequency */
317     SystemCoreClock »= tmp;
318 }
319
320 /**
321 * @}

```

4.17.2.2 SystemInit()

```
void SystemInit (
    void )
```

Setup the microcontroller system. Initialize the default HSI clock source, vector table location and the PLL configuration is reset.

Parameters

<i>None</i>	<input type="button" value=""/>
-------------	---------------------------------

Return values

<i>None</i>	<input type="button" value=""/>
-------------	---------------------------------

Definition at line 166 of file system_stm32f0xx.c.

261 {

Chapter 5

Class Documentation

5.1 adc_command Union Reference

```
#include <adc_controller.h>
```

Collaboration diagram for adc_command:

adc_command
+ value + b1 + b2 + bytes + FIFO_TRIGGER_LEVEL + EOC_INTERRUPT_FUNC + SWEEP_SEQ_SELECT + CONVERSION_MODE_SELECT + INPUT_SELECT_MODE + CLK_SOURCE + SAMPLE_PERIOD + CONV_OUTPUT_FORMAT + D11 + CMR + bits

Public Attributes

- uint16_t `value`

- struct {
 uint8_t b1
 uint8_t b2
} bytes

- struct {
 uint8_t FIFO_TRIG_LEVEL: 2
 uint8_t EOC_INT_FUNC: 1
 uint8_t SWEEP_SEQ_SELECT: 2
 uint8_t CONVERSION_MODE_SELECT: 2
 uint8_t INPUT_SELECT_MODE: 1
 uint8_t CLK_SOURCE: 1
 uint8_t SAMPLE_PERIOD: 1
 uint8_t CONV_OUTPUT_FORMAT: 1
 uint8_t D11: 1
 uint8_t CMR: 4
} bits

5.1.1 Detailed Description

Definition at line 6 of file adc_controller.h.

5.1.2 Member Data Documentation

5.1.2.1 b1

```
uint8_t adc_command::b1
```

Definition at line 14 of file adc_controller.h.

5.1.2.2 b2

```
uint8_t adc_command::b2
```

Definition at line 15 of file adc_controller.h.

5.1.2.3 bits

```
struct { ... } adc_command::bits
```

5.1.2.4 bytes

```
struct { ... } adc_command::bytes
```

5.1.2.5 CLK_SOURCE

```
uint8_t adc_command::CLK_SOURCE
```

Definition at line 45 of file adc_controller.h.

5.1.2.6 CMR

```
uint8_t adc_command::CMR
```

Definition at line 57 of file adc_controller.h.

5.1.2.7 CONV_OUTPUT_FORMAT

```
uint8_t adc_command::CONV_OUTPUT_FORMAT
```

Definition at line 53 of file adc_controller.h.

5.1.2.8 CONVERSION_MODE_SELECT

```
uint8_t adc_command::CONVERSION_MODE_SELECT
```

Definition at line 37 of file adc_controller.h.

5.1.2.9 D11

```
uint8_t adc_command::D11
```

Definition at line 55 of file adc_controller.h.

5.1.2.10 EOC_INT_FUNC

```
uint8_t adc_command::EOC_INT_FUNC
```

Definition at line 26 of file adc_controller.h.

5.1.2.11 FIFO_TRIG_LEVEL

```
uint8_t adc_command::FIFO_TRIG_LEVEL
```

Definition at line 23 of file adc_controller.h.

5.1.2.12 INPUT_SELECT_MODE

```
uint8_t adc_command::INPUT_SELECT_MODE
```

Definition at line 41 of file adc_controller.h.

5.1.2.13 SAMPLE_PERIOD

```
uint8_t adc_command::SAMPLE_PERIOD
```

Definition at line 49 of file adc_controller.h.

5.1.2.14 SWEEP_SEQ_SELECT

```
uint8_t adc_command::SWEEP_SEQ_SELECT
```

Definition at line 31 of file adc_controller.h.

5.1.2.15 value

```
uint16_t adc_command::value
```

Definition at line 11 of file adc_controller.h.

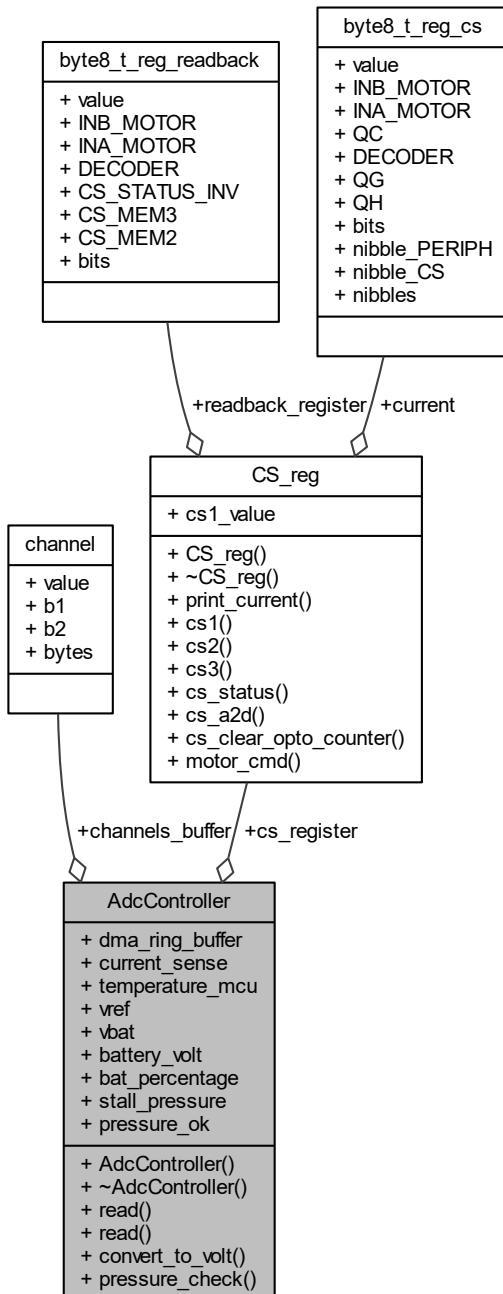
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[adc_controller.h](#)

5.2 AdcController Class Reference

```
#include <adc_controller.h>
```

Collaboration diagram for AdcController:



Public Member Functions

- [AdcController \(CS_reg *\)](#)

- `~AdcController ()`
- `void read (enum select_channel)`
- `void read (void)`
- `uint32_t convert_to_volt (void)`
- `void pressure_check (void)`

Public Attributes

- `uint32_t dma_ring_buffer [DMA_LENGTH]`
- `channel channels_buffer [8]`
- `uint32_t current_sense`
- `uint32_t temperature_mcu`
- `uint32_t vref`
- `uint32_t vbat`
- `CS_reg * cs_register`
- `uint32_t battery_volt`
- `uint32_t bat_percentage`
- `bool stall_pressure`
- `bool pressure_ok`

5.2.1 Detailed Description

Definition at line 86 of file adc_controller.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 AdcController()

```
AdcController::AdcController (
    CS_reg * cs_reg )
```

Definition at line 5 of file adc_controller.cpp.

```
6 {
7     this->cs_register = cs_reg;
8     this->cfr.bits.D11 = 1;
9     this->cfr.bits.CMR = 0b1000;
10 }
```

5.2.2.2 ~AdcController()

```
AdcController::~AdcController ( )
```

Definition at line 12 of file adc_controller.cpp.

```
13 {
14
15 }
```

5.2.3 Member Function Documentation

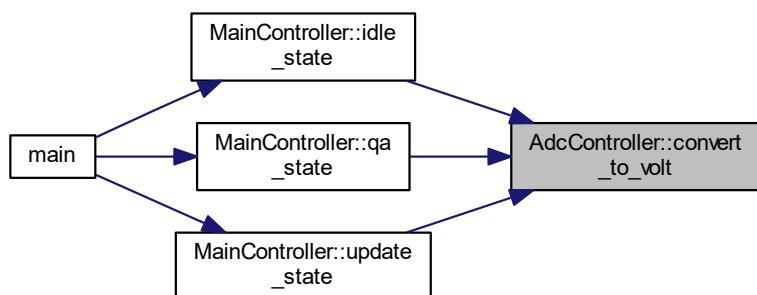
5.2.3.1 convert_to_volt()

```
uint32_t AdcController::convert_to_volt (
    void )
```

Definition at line 68 of file adc_controller.cpp.

```
69 {
70     // This function gives volt in milivolts
71     this->battery_volt = (this->channels_buffer[V12].value * 7 + (channels_buffer[V12].value * 5) / 10 +
8552);
72     this->bat_percentage = (this->battery_volt - 9500) / 30;
73     if (this->battery_volt < 9500)
74     {
75         this->bat_percentage = 0;
76     }
77     else if (this->battery_volt > 12500)
78     {
79         this->bat_percentage = 100;
80     }
81     return this->bat_percentage;
82 }
```

Here is the caller graph for this function:



5.2.3.2 pressure_check()

```
void AdcController::pressure_check (
    void )
```

Definition at line 84 of file adc_controller.cpp.

```
85 {
86     if (this->channels_buffer[Pressure].value > ADC_Presure_27_bar)
87     {
88         this->pressure_ok = true;
89         this->stall_pressure = false;
90     }
91     else if (this->channels_buffer[Pressure].value < ADC_Presure_20_bar)
92     {
93         this->pressure_ok = false;
```

```

94         this->stall_pressure = true;
95     }
96     else
97     {
98         this->pressure_ok = false;
99         this->stall_pressure = false;
100    }
101 }

```

5.2.3.3 `read()` [1/2]

```
void AdcController::read (
    enum select_channel channel_ )
```

Definition at line 45 of file adc_controller.cpp.

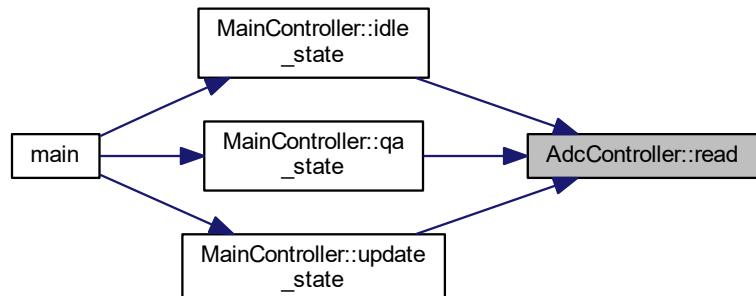
```

46 {
47     this->cs_register->cs_a2d(LOW);
48     uint8_t txBuffer[3];
49     uint8_t rxBuffer[3];
50     txBuffer[0] = 0x01; // Start Bit
51     txBuffer[1] = ((static_cast<uint8_t>(channel_) << 3) & 0xf0) | 0x80;
52     txBuffer[2] = 0x0;
53     HAL_SPI_TransmitReceive(&hspil, txBuffer, rxBuffer, 3,10);
54     uint16_t adc_value= (((uint16_t)rxBuffer[1] & 0x07)<<8) + ((uint16_t)rxBuffer[2] & 0xff);
55     this->channels_buffer[channel_].value = adc_value;
56     this->cs_register->cs_a2d(HIGH);
57 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.2.3.4 `read()` [2/2]

```
void AdcController::read (
    void )
```

Definition at line 35 of file adc_controller.cpp.

```
36 {
37     for (uint8_t i = 0; i < 8; i++)
38     {
39         enum select_channel channel_ = static_cast<select_channel>(i);
40         this->read(channel_);
41         HAL_Delay(1);
42     }
43 }
```

5.2.4 Member Data Documentation

5.2.4.1 `bat_percentage`

```
uint32_t AdcController::bat_percentage
```

Definition at line 97 of file adc_controller.h.

5.2.4.2 `battery_volt`

```
uint32_t AdcController::battery_volt
```

Definition at line 96 of file adc_controller.h.

5.2.4.3 `channels_buffer`

```
channel AdcController::channels_buffer[8]
```

Definition at line 90 of file adc_controller.h.

5.2.4.4 `cs_register`

```
CS_Reg* AdcController::cs_register
```

Definition at line 95 of file adc_controller.h.

5.2.4.5 **current_sense**

```
uint32_t AdcController::current_sense
```

Definition at line 91 of file adc_controller.h.

5.2.4.6 **dma_ring_buffer**

```
uint32_t AdcController::dma_ring_buffer[DMA_LENGTH]
```

Definition at line 89 of file adc_controller.h.

5.2.4.7 **pressure_ok**

```
bool AdcController::pressure_ok
```

Definition at line 99 of file adc_controller.h.

5.2.4.8 **stall_pressure**

```
bool AdcController::stall_pressure
```

Definition at line 98 of file adc_controller.h.

5.2.4.9 **temperature_mcu**

```
uint32_t AdcController::temperature_mcu
```

Definition at line 92 of file adc_controller.h.

5.2.4.10 **vbat**

```
uint32_t AdcController::vbat
```

Definition at line 94 of file adc_controller.h.

5.2.4.11 vref

```
uint32_t AdcController::vref
```

Definition at line 93 of file adc_controller.h.

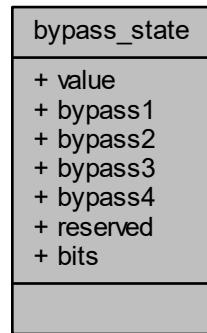
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc_controller.h
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc_controller.cpp

5.3 bypass_state Union Reference

```
#include <state.h>
```

Collaboration diagram for bypass_state:



Public Attributes

- uint8_t **value**
- struct {
 - uint8_t **bypass1**: 1
 - uint8_t **bypass2**: 1
 - uint8_t **bypass3**: 1
 - uint8_t **bypass4**: 1
 - uint8_t **reserved**: 4}
- } **bits**

5.3.1 Detailed Description

Definition at line 18 of file state.h.

5.3.2 Member Data Documentation

5.3.2.1 bits

```
struct { ... } bypass_state::bits
```

5.3.2.2 bypass1

```
uint8_t bypass_state::bypass1
```

Definition at line 24 of file state.h.

5.3.2.3 bypass2

```
uint8_t bypass_state::bypass2
```

Definition at line 25 of file state.h.

5.3.2.4 bypass3

```
uint8_t bypass_state::bypass3
```

Definition at line 26 of file state.h.

5.3.2.5 bypass4

```
uint8_t bypass_state::bypass4
```

Definition at line 27 of file state.h.

5.3.2.6 reserved

```
uint8_t bypass_state::reserved
```

Definition at line 28 of file state.h.

5.3.2.7 value

```
uint8_t bypass_state::value
```

Definition at line 20 of file state.h.

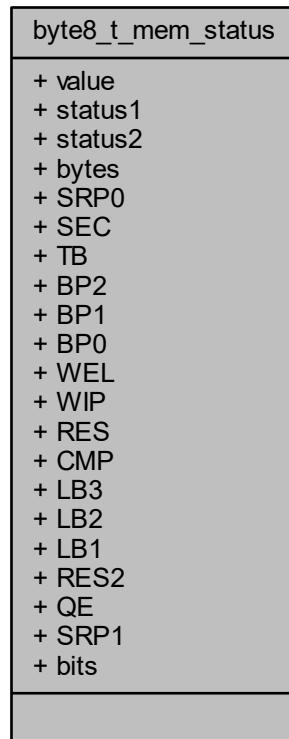
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/state.h

5.4 byte8_t_mem_status Union Reference

```
#include <mem.h>
```

Collaboration diagram for byte8_t_mem_status:



Public Attributes

- `uint16_t value`
- `struct {`
 - `uint8_t status1`
 - `uint8_t status2``} bytes`
- `struct {`
 - `uint8_t SRP0: 1`
 - `uint8_t SEC: 1`
 - `uint8_t TB: 1`
 - `uint8_t BP2: 1`
 - `uint8_t BP1: 1`
 - `uint8_t BP0: 1`
 - `uint8_t WEL: 1`
 - `uint8_t WIP: 1`
 - `uint8_t RES: 1`
 - `uint8_t CMP: 1`
 - `uint8_t LB3: 1`
 - `uint8_t LB2: 1`
 - `uint8_t LB1: 1`
 - `uint8_t RES2: 1`
 - `uint8_t QE: 1`
 - `uint8_t SRP1: 1``} bits`

5.4.1 Detailed Description

Definition at line 6 of file mem.h.

5.4.2 Member Data Documentation

5.4.2.1 bits

```
struct { ... } byte8_t_mem_status::bits
```

5.4.2.2 BP0

```
uint8_t byte8_t_mem_status::BP0
```

Definition at line 24 of file mem.h.

5.4.2.3 BP1

```
uint8_t byte8_t_mem_status::BP1
```

Definition at line 23 of file mem.h.

5.4.2.4 BP2

```
uint8_t byte8_t_mem_status::BP2
```

Definition at line 22 of file mem.h.

5.4.2.5 bytes

```
struct { ... } byte8_t_mem_status::bytes
```

5.4.2.6 CMP

```
uint8_t byte8_t_mem_status::CMP
```

Definition at line 28 of file mem.h.

5.4.2.7 LB1

```
uint8_t byte8_t_mem_status::LB1
```

Definition at line 31 of file mem.h.

5.4.2.8 LB2

```
uint8_t byte8_t_mem_status::LB2
```

Definition at line 30 of file mem.h.

5.4.2.9 LB3

```
uint8_t byte8_t_mem_status::LB3
```

Definition at line 29 of file mem.h.

5.4.2.10 QE

```
uint8_t byte8_t_mem_status::QE
```

Definition at line 33 of file mem.h.

5.4.2.11 RES

```
uint8_t byte8_t_mem_status::RES
```

Definition at line 27 of file mem.h.

5.4.2.12 RES2

```
uint8_t byte8_t_mem_status::RES2
```

Definition at line 32 of file mem.h.

5.4.2.13 SEC

```
uint8_t byte8_t_mem_status::SEC
```

Definition at line 20 of file mem.h.

5.4.2.14 SRP0

```
uint8_t byte8_t_mem_status::SRP0
```

Definition at line 19 of file mem.h.

5.4.2.15 SRP1

```
uint8_t byte8_t_mem_status::SRP1
```

Definition at line 34 of file mem.h.

5.4.2.16 status1

```
uint8_t byte8_t_mem_status::status1
```

Definition at line 14 of file mem.h.

5.4.2.17 status2

```
uint8_t byte8_t_mem_status::status2
```

Definition at line 15 of file mem.h.

5.4.2.18 TB

```
uint8_t byte8_t_mem_status::TB
```

Definition at line 21 of file mem.h.

5.4.2.19 value

```
uint16_t byte8_t_mem_status::value
```

Definition at line 11 of file mem.h.

5.4.2.20 WEL

```
uint8_t byte8_t_mem_status::WEL
```

Definition at line 25 of file mem.h.

5.4.2.21 WIP

```
uint8_t byte8_t_mem_status::WIP
```

Definition at line 26 of file mem.h.

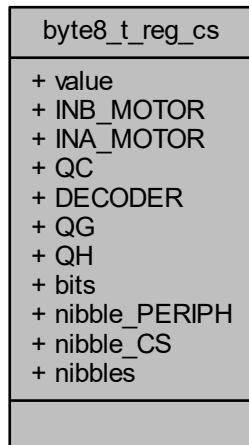
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem.h

5.5 byte8_t_reg_cs Union Reference

```
#include <cs_register.h>
```

Collaboration diagram for byte8_t_reg_cs:



Public Attributes

- uint8_t **value**
- struct {
 - uint8_t **INB_MOTOR**: 1
 - uint8_t **INA_MOTOR**: 1
 - uint8_t **QC**: 1
 - uint8_t **DECODER**: 3
 - uint8_t **QG**: 1
 - uint8_t **QH**: 1
} **bits**
- struct {
 - uint8_t **nibble_PERIPH**: 4
 - uint8_t **nibble_CS**: 4
} **nibbles**

5.5.1 Detailed Description

Definition at line 5 of file cs_register.h.

5.5.2 Member Data Documentation

5.5.2.1 bits

```
struct { ... } byte8_t_reg_cs::bits
```

5.5.2.2 DECODER

```
uint8_t byte8_t_reg_cs::DECODER
```

Definition at line 16 of file cs_register.h.

5.5.2.3 INA_MOTOR

```
uint8_t byte8_t_reg_cs::INA_MOTOR
```

Definition at line 14 of file cs_register.h.

5.5.2.4 INB_MOTOR

```
uint8_t byte8_t_reg_cs::INB_MOTOR
```

Definition at line 13 of file cs_register.h.

5.5.2.5 nibble_CS

```
uint8_t byte8_t_reg_cs::nibble_CS
```

Definition at line 24 of file cs_register.h.

5.5.2.6 nibble_PERIPH

```
uint8_t byte8_t_reg_cs::nibble_PERIPH
```

Definition at line 23 of file cs_register.h.

5.5.2.7 nibbles

```
struct { ... } byte8_t_reg_cs::nibbles
```

5.5.2.8 QC

```
uint8_t byte8_t_reg_cs::QC
```

Definition at line 15 of file cs_register.h.

5.5.2.9 QG

```
uint8_t byte8_t_reg_cs::QG
```

Definition at line 17 of file cs_register.h.

5.5.2.10 QH

```
uint8_t byte8_t_reg_cs::QH
```

Definition at line 18 of file cs_register.h.

5.5.2.11 value

```
uint8_t byte8_t_reg_cs::value
```

Definition at line 10 of file cs_register.h.

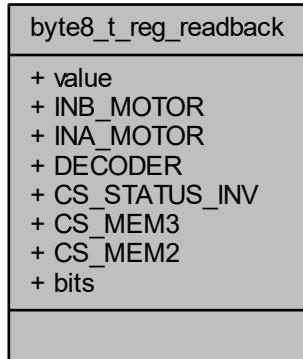
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h

5.6 byte8_t_reg_readback Union Reference

```
#include <cs_register.h>
```

Collaboration diagram for byte8_t_reg_readback:



Public Attributes

- uint8_t `value`
- struct {
 - uint8_t `INB_MOTOR`: 1
 - uint8_t `INA_MOTOR`: 1
 - uint8_t `DECODER`: 3
 - uint8_t `CS_STATUS_INV`: 1
 - uint8_t `CS_MEM3`: 1
 - uint8_t `CS_MEM2`: 1}
- } `bits`

5.6.1 Detailed Description

Definition at line 29 of file `cs_register.h`.

5.6.2 Member Data Documentation

5.6.2.1 bits

```
struct { ... } byte8_t_reg_readback::bits
```

5.6.2.2 CS_MEM2

```
uint8_t byte8_t_reg_readback::CS_MEM2
```

Definition at line 42 of file cs_register.h.

5.6.2.3 CS_MEM3

```
uint8_t byte8_t_reg_readback::CS_MEM3
```

Definition at line 41 of file cs_register.h.

5.6.2.4 CS_STATUS_INV

```
uint8_t byte8_t_reg_readback::CS_STATUS_INV
```

Definition at line 40 of file cs_register.h.

5.6.2.5 DECODER

```
uint8_t byte8_t_reg_readback::DECODER
```

Definition at line 39 of file cs_register.h.

5.6.2.6 INA_MOTOR

```
uint8_t byte8_t_reg_readback::INA_MOTOR
```

Definition at line 38 of file cs_register.h.

5.6.2.7 INB_MOTOR

```
uint8_t byte8_t_reg_readback::INB_MOTOR
```

Definition at line 37 of file cs_register.h.

5.6.2.8 value

```
uint8_t byte8_t_reg_readback::value
```

Definition at line 34 of file cs_register.h.

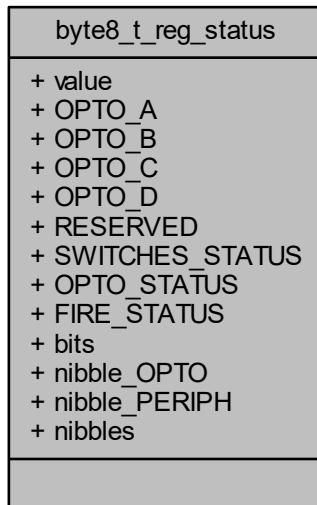
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h

5.7 byte8_t_reg_status Union Reference

```
#include <status_reg.h>
```

Collaboration diagram for byte8_t_reg_status:



Public Attributes

- uint8_t **value**
- struct {
 - uint8_t **OPTO_A**: 1
 - uint8_t **OPTO_B**: 1
 - uint8_t **OPTO_C**: 1
 - uint8_t **OPTO_D**: 1
 - uint8_t **RESERVED**: 1
 - uint8_t **SWITCHES_STATUS**: 1
 - uint8_t **OPTO_STATUS**: 1
 - uint8_t **FIRE_STATUS**: 1}
- struct {
 - uint8_t **nibble_OPTO**: 4
 - uint8_t **nibble_PERIPH**: 4}
- **nibbles**

5.7.1 Detailed Description

Definition at line 4 of file status_reg.h.

5.7.2 Member Data Documentation

5.7.2.1 bits

```
struct { ... } byte8_t_reg_status::bits
```

5.7.2.2 FIRE_STATUS

```
uint8_t byte8_t_reg_status::FIRE_STATUS
```

Definition at line 19 of file status_reg.h.

5.7.2.3 nibble_OPTO

```
uint8_t byte8_t_reg_status::nibble_OPTO
```

Definition at line 24 of file status_reg.h.

5.7.2.4 nibble_PERIPH

```
uint8_t byte8_t_reg_status::nibble_PERIPH
```

Definition at line 25 of file status_reg.h.

5.7.2.5 nibbles

```
struct { ... } byte8_t_reg_status::nibbles
```

5.7.2.6 OPTO_A

```
uint8_t byte8_t_reg_status::OPTO_A
```

Definition at line 12 of file status_reg.h.

5.7.2.7 OPTO_B

```
uint8_t byte8_t_reg_status::OPTO_B
```

Definition at line 13 of file status_reg.h.

5.7.2.8 OPTO_C

```
uint8_t byte8_t_reg_status::OPTO_C
```

Definition at line 14 of file status_reg.h.

5.7.2.9 OPTO_D

```
uint8_t byte8_t_reg_status::OPTO_D
```

Definition at line 15 of file status_reg.h.

5.7.2.10 OPTO_STATUS

```
uint8_t byte8_t_reg_status::OPTO_STATUS
```

Definition at line 18 of file status_reg.h.

5.7.2.11 RESERVED

```
uint8_t byte8_t_reg_status::RESERVED
```

Definition at line 16 of file status_reg.h.

5.7.2.12 SWITCHES_STATUS

```
uint8_t byte8_t_reg_status::SWITCHES_STATUS
```

Definition at line 17 of file status_reg.h.

5.7.2.13 value

```
uint8_t byte8_t_reg_status::value
```

Definition at line 9 of file status_reg.h.

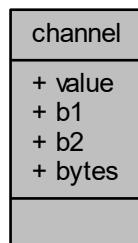
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[status_reg.h](#)

5.8 channel Union Reference

```
#include <adc_controller.h>
```

Collaboration diagram for channel:



Public Attributes

- int16_t [value](#)
- struct {
 uint8_t [b1](#)
 uint8_t [b2](#)
} [bytes](#)

5.8.1 Detailed Description

Definition at line 61 of file adc_controller.h.

5.8.2 Member Data Documentation

5.8.2.1 b1

```
uint8_t channel::b1
```

Definition at line 69 of file adc_controller.h.

5.8.2.2 b2

```
uint8_t channel::b2
```

Definition at line 70 of file adc_controller.h.

5.8.2.3 bytes

```
struct { ... } channel::bytes
```

5.8.2.4 value

```
int16_t channel::value
```

Definition at line 66 of file adc_controller.h.

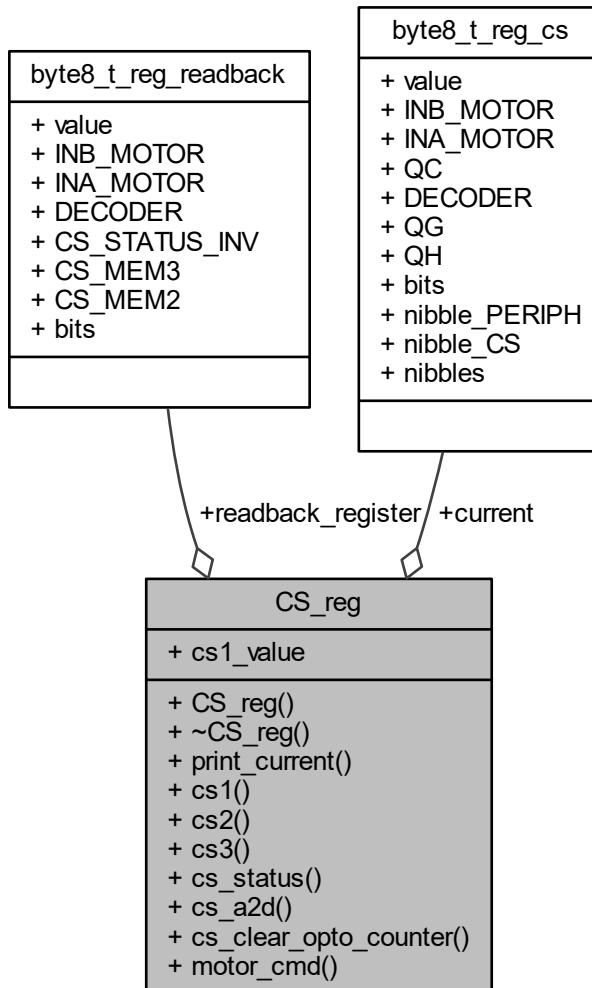
The documentation for this union was generated from the following file:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc_controller.h

5.9 CS_reg Class Reference

```
#include <cs_register.h>
```

Collaboration diagram for CS_reg:



Public Member Functions

- `CS_reg ()`
- `~CS_reg ()`
- `void print_current (void)`
- `void cs1 (GPIO_PinState)`
- `void cs2 (GPIO_PinState)`
- `void cs3 (GPIO_PinState)`
- `void cs_status (GPIO_PinState)`
- `void cs_a2d (GPIO_PinState)`
- `void cs_clear_opto_counter (GPIO_PinState status)`
- `void motor_cmd (bool state)`

Public Attributes

- `byte8_t reg_current`
- `byte8_t reg_readback readback_register`
- `bool cs1_value`

5.9.1 Detailed Description

Definition at line 57 of file `cs_register.h`.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 CS_reg()

```
CS_reg::CS_reg ( )
```

Definition at line 8 of file `cs_register.cpp`.

```
9 {
10     this->current.value = 0;
11 }
```

5.9.2.2 ~CS_reg()

```
CS_reg::~CS_reg ( )
```

Definition at line 13 of file `cs_register.cpp`.

```
14 {
15
16 }
```

5.9.3 Member Function Documentation

5.9.3.1 cs1()

```
void CS_reg::cs1 (
    GPIO_PinState status )
```

Definition at line 28 of file `cs_register.cpp`.

```
29 {
30
31     HAL_GPIO_WritePin(CHIP_SELECT MCU_MEM_GPIO_Port, CHIP_SELECT MCU_MEM_Pin, status);
32     this->cs1_value = (bool)status;
33 }
```

5.9.3.2 cs2()

```
void CS_reg::cs2 (
    GPIO_PinState status )
```

IMPORTANT: The usual "Hardware logic" is chipselect is active on low. But since we have a special case where the cs_register is reset to 0 We have inverted all cs inputs to the hardware from cs_register so what needs to be done is that the register written logical 1 to get an electrical VLow

Definition at line 35 of file cs_register.cpp.

```
36 {
37     /*****
38     * IMPORTANT: The usual "Hardware logic" is chipselect is active on low.
39     * But since we have a special case where the cs_register is reset to 0
40     * We have inverted all cs inputs to the hardware from cs_register
41     * so what needs to be done is that the register written logical 1
42     * to get an electrical VLow
43     *****/
44     enum cs_decoder picked_cs = Y0;
45     if (!(bool)status)
46     {
47         picked_cs = CS_MEM2;
48     }
49     this->current.bits.DECODER = picked_cs;
50     HAL_SPI_Transmit(&hspil, &this->current.value, 1, 10);
51     this->latch_register(HIGH);
52     this->store_register();
```

5.9.3.3 cs3()

```
void CS_reg::cs3 (
    GPIO_PinState status )
```

Definition at line 54 of file cs_register.cpp.

```
56 {
57     enum cs_decoder picked_cs = Y0;
58     if (!(bool)status)
59     {
60         picked_cs = CS_MEM3;
61     }
62     this->current.bits.DECODER = picked_cs;
63     HAL_SPI_Transmit(&hspil, &this->current.value, 1, 10);
64     this->latch_register(HIGH);
65     this->store_register();
```

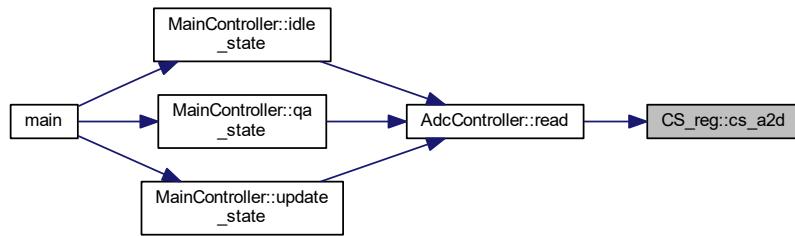
5.9.3.4 cs_a2d()

```
void CS_reg::cs_a2d (
    GPIO_PinState status )
```

Definition at line 80 of file cs_register.cpp.

```
82 {
83     enum cs_decoder picked_cs = Y0;
84     if (!(bool)status)
85     {
86         picked_cs = CS_A2D;
87     }
88     this->current.bits.DECODER = picked_cs;
89     HAL_SPI_Transmit(&hspil, &this->current.value, 1, 10);
90     this->latch_register(HIGH);
91     this->store_register();
```

Here is the caller graph for this function:



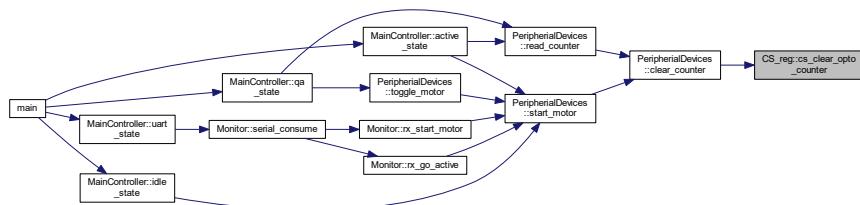
5.9.3.5 cs_clear_opto_counter()

```
void CS_reg::cs_clear_opto_counter (
    GPIO_PinState status )
```

Definition at line 93 of file cs_register.cpp.

```
95 {
96     enum cs_decoder picked_cs = Y0;
97     if (!(bool)status)
98     {
99         picked_cs = OPTO_CNT_CLRN;
100    }
101    this->current.bits.DECODER = picked_cs;
102    HAL_SPI_Transmit(&hspil, &this->current.value, 1, 10);
103    this->latch_register(HIGH);
104    this->store_register();
```

Here is the caller graph for this function:



5.9.3.6 cs_status()

```
void CS_reg::cs_status (
    GPIO_PinState status )
```

Definition at line 67 of file cs_register.cpp.

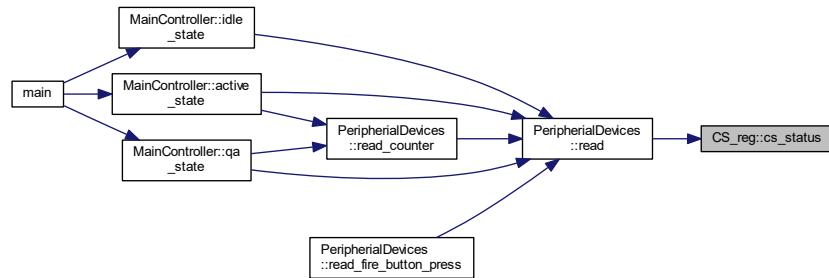
```
69 {
70     enum cs_decoder picked_cs = Y0;
```

```

71     if (!bool)status)
72     {
73         picked_cs = CS_STATUS;
74     }
75     this->current.bits.DECODER = picked_cs;
76     HAL_SPI_Transmit(&hspl1, &this->current.value, 1, 10);
77     this->latch_register(HIGH);
78     this->store_register();

```

Here is the caller graph for this function:



5.9.3.7 motor_cmd()

```

void CS_reg::motor_cmd (
    bool state )

```

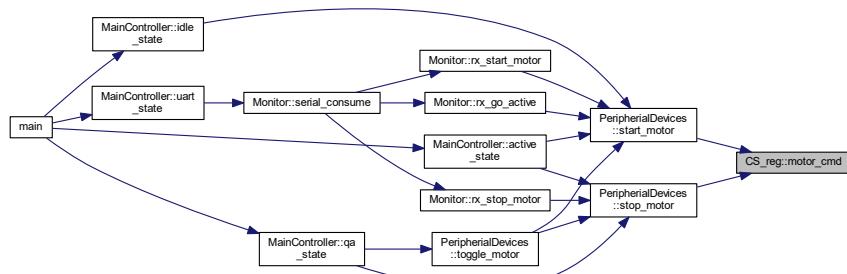
Definition at line 164 of file cs_register.cpp.

```

166 {
167     if (state)
168     {
169         this->current.bits.INA_MOTOR = 1;
170         this->current.bits.INB_MOTOR = 0;
171     }
172     else
173     {
174         this->current.bits.INA_MOTOR = 0;
175         this->current.bits.INB_MOTOR = 0;
176     }
177     uint8_t value = current.value;
178     this->set_value(value);

```

Here is the caller graph for this function:



5.9.3.8 print_current()

```
void CS_reg::print_current (
    void )
```

Definition at line 106 of file cs_register.cpp.

```
108 {
109     uint8_t status[128];
110     sprintf((char*)status, "[%d] INB_MOTOR\r\n\"\
111         \"%[d] INA_MOTOR\r\n\"\
112         \"%[08x] DECODER\r\n\",\
113         this->current.bits.INB_MOTOR,
114         this->current.bits.INA_MOTOR,
115         this->current.bits.DECODER);
116     // TODO: consider decorator
117     HAL_UART_Transmit(SERIAL_PC, status, strlen((char*)status), 100);
```

5.9.4 Member Data Documentation

5.9.4.1 cs1_value

```
bool CS_reg::cs1_value
```

Definition at line 63 of file cs_register.h.

5.9.4.2 current

```
byte8_t_reg_CS CS_reg::current
```

Definition at line 61 of file cs_register.h.

5.9.4.3 readback_register

```
byte8_t_reg_Readback CS_reg::readback_register
```

Definition at line 62 of file cs_register.h.

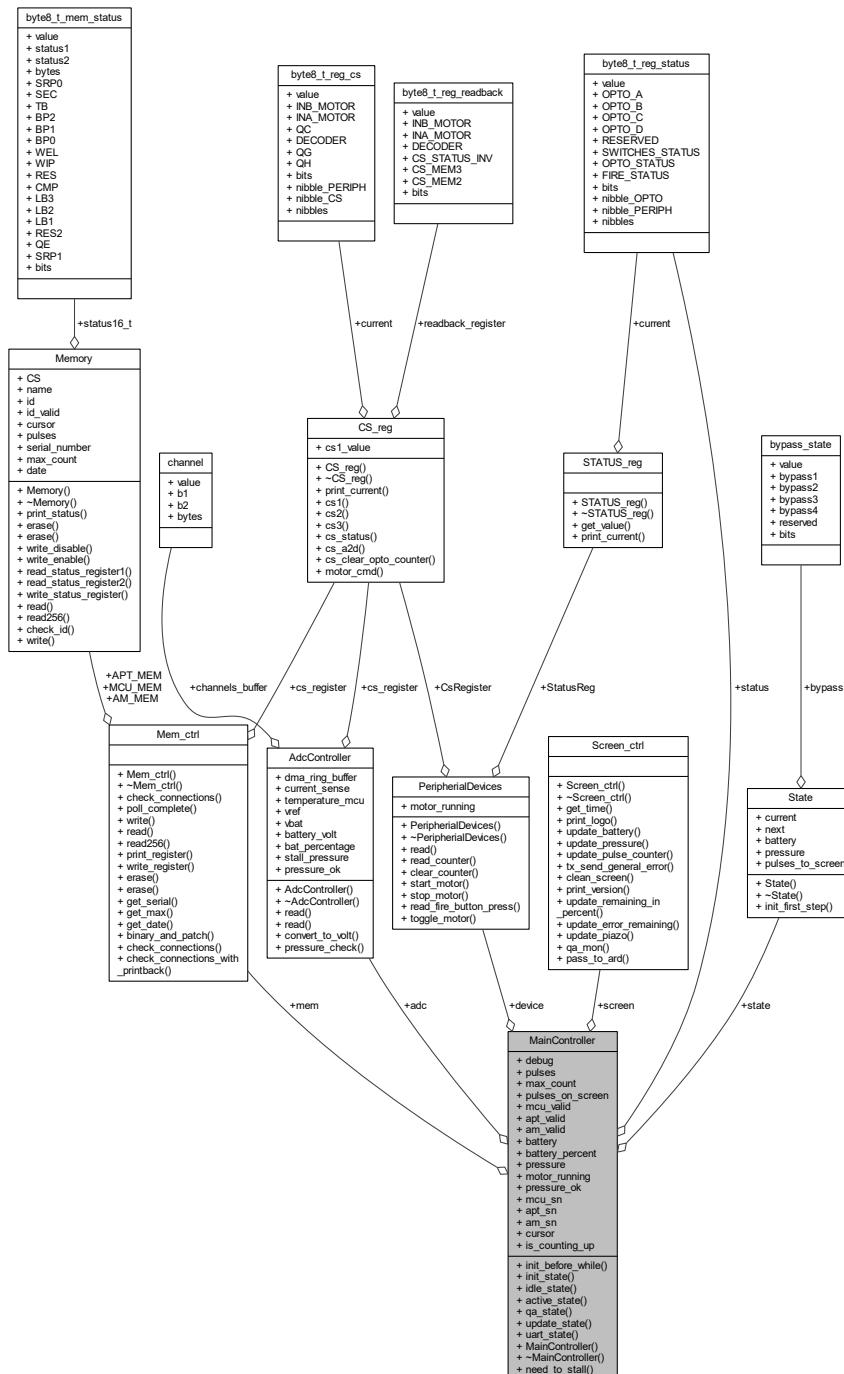
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/cs_register.cpp

5.10 MainController Class Reference

```
#include <main_controller.h>
```

Collaboration diagram for MainController:



Public Member Functions

- void [init_before_while\(\)](#)

- void `init_state ()`
- void `idle_state ()`
- void `active_state ()`
- void `qa_state ()`
- void `update_state ()`
- void `uart_state ()`
- `MainController (DebugHelper *, Mem_ctrl *, PeripheralDevices *, State *, Screen_ctrl *, AdcController *)`
- `~MainController ()`
- bool `need_to_stall ()`

Public Attributes

- `DebugHelper * debug`
- `Mem_ctrl * mem`
- `PeripheralDevices * device`
- `State * state`
- `Screen_ctrl * screen`
- `AdcController * adc`
- `uint32_t * pulses`
- `uint32_t * max_count`
- `uint32_t pulses_on_screen`
- `bool * mcu_valid`
- `bool * apt_valid`
- `bool * am_valid`
- `int16_t * battery`
- `uint32_t * battery_percent`
- `int16_t * pressure`
- `bool * motor_running`
- `bool * pressure_ok`
- `byte8_t_reg_status * status`
- `uint32_t * mcu_sn`
- `uint32_t * apt_sn`
- `uint32_t * am_sn`
- `uint32_t * cursor`
- `bool is_counting_up`

5.10.1 Detailed Description

Definition at line 9 of file main_controller.h.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 MainController()

```
MainController::MainController (
    DebugHelper * debug_,
    Mem_ctrl * mem_,
    PeripheralDevices * device_,
    State * state_,
    Screen_ctrl * screen_,
    AdcController * adc_ )
```

Definition at line 205 of file main_controller.cpp.

```
206 {
207     this->debug = debug_;
208     this->mem = mem_;
209     this->device = device_;
210     this->state = state_;
211     this->screen = screen_;
212     this->adc = adc_;
213
214     // pointers in controller
215     //memories:
216     this->mcu_valid = &this->mem->MCU_MEM->id_valid;
217     this->apt_valid = &this->mem->APT_MEM->id_valid;
218     this->am_valid = &this->mem->AM_MEM->id_valid;
219     this->mcu_sn = &this->mem->MCU_MEM->serial_number;
220     this->apt_sn = &this->mem->APT_MEM->serial_number;
221     this->am_sn = &this->mem->AM_MEM->serial_number;
222     this->max_count = &this->mem->AM_MEM->max_count;
223     this->pulses = &this->mem->AM_MEM->pulses;
224     this->cursor = &this->mem->AM_MEM->cursor;
225
226     //status:
227     this->motor_running = &this->device->motor_running;
228     this->status = &this->device->StatusReg->current;
229     this->pulses_on_screen = 0;
230     this->is_counting_up = true;
231
232     //adc
233     this->battery = &this->adc->channels_buffer[V12].value;
234     this->battery_percent = &this->adc->bat_percentage;
235     this->pressure = &this->adc->channels_buffer[Pressure].value;
236     this->pressure_ok = &this->adc->pressure_ok;
237 }
```

5.10.2.2 ~MainController()

```
MainController::~MainController ( )
```

Definition at line 248 of file main_controller.cpp.

```
249 {
250
251 }
```

5.10.3 Member Function Documentation

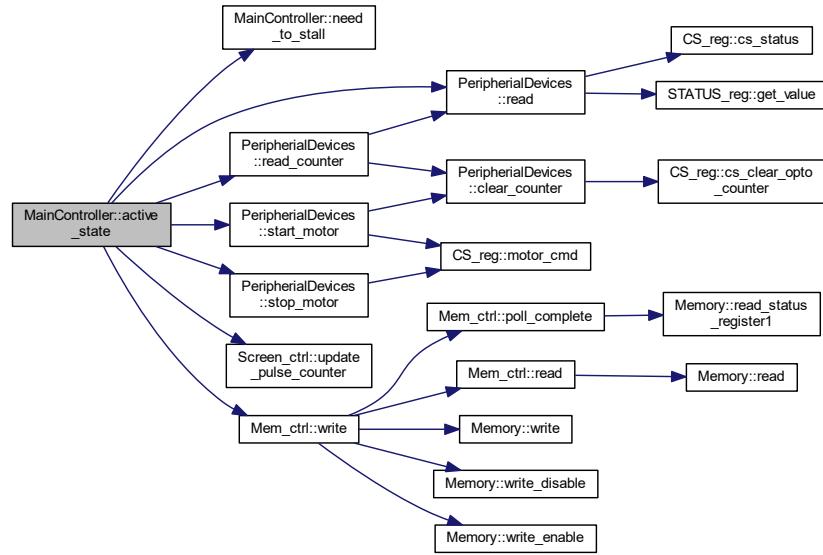
5.10.3.1 active_state()

```
void MainController::active_state ( )
```

Definition at line 102 of file main_controller.cpp.

```
103 {
104     this->device->read();
105     if (this->need_to_stall())
106     {
107         this->device->stop_motor();
108         this->state->next = Idle;
109     }
110     if (this->device->StatusReg->current.bits.FIRE_STATUS)
111     {
112         if (!this->motor_running && !need_to_stall())
113         {
114             this->device->start_motor();
115             this->state->next = Active;
116         }
117         else
118         {
119             this->state->next = Idle;
120         }
121     }
122     volatile uint32_t added_pulses = this->device->read_counter();
123     if (added_pulses > 0)
124     {
125         for (uint32_t i = 1; i <= added_pulses; i++)
126         {
127             this->mem->write(3, *this->cursor + 8*i - 4, (*this->pulses) + 1);
128             this->mem->write(3, *this->cursor + 8*i, (*this->pulses) + 1);
129         }
130         this->mem->AM_MEM->pulses += added_pulses; // add a function to spi write and increment
131         this->mem->AM_MEM->cursor += added_pulses * 8;
132         if (this->is_counting_up)
133         {
134             this->pulses_on_screen += added_pulses;
135         }
136         else
137         {
138             if (this->pulses_on_screen > added_pulses)
139             {
140                 this->pulses_on_screen = this->pulses_on_screen - added_pulses;
141             }
142             else
143             {
144                 this->pulses_on_screen = 0;
145                 this->device->stop_motor();
146             }
147         }
148     }
149     this->screen->update_pulse_counter(this->pulses_on_screen);
150 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.3.2 idle_state()

```
void MainController::idle_state ( )
```

Definition at line 65 of file main_controller.cpp.

```

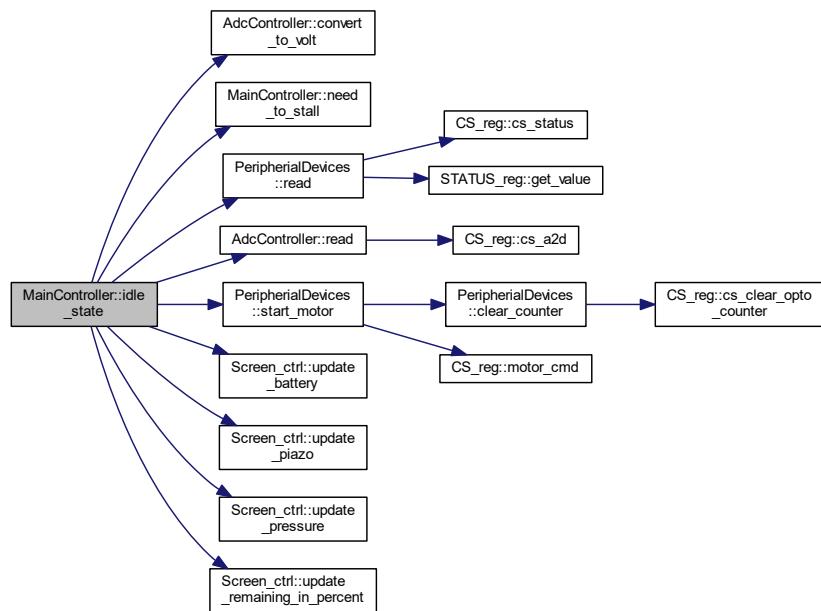
66 {
67     if (reset_pressed)
68     {
69         this->is_counting_up = this->is_counting_up;
70         if (this->is_counting_up)
71         {
72             this->pulses_on_screen = 0;
73         }
74         else
75         {
76             this->pulses_on_screen = 400;
77         }
78         reset_pressed = false;
79     }
80     this->adc->read();
  
```

```

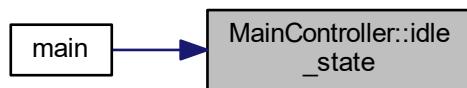
81     this->adc->convert_to_volt();
82     this->device->read();
83     if (this->device->StatusReg->current.bits.FIRE_STATUS)
84     {
85         if (!this->motor_running && !need_to_stall())
86         {
87             this->device->start_motor();
88             this->state->next = Active;
89         }
90     }
91     else
92     {
93         this->state->next = Idle;
94     }
95     this->screen->update_battery(*this->battery, *this->battery_percent);
96     this->screen->update_pressure(*this->pressure);
97     this->screen->update_remaining_in_percent(*this->max_count, *this->pulses);
98     this->screen->update_piazo(*this->motor_running);
99     this->debug->update(*this->motor_running, *this->pressure_ok, *this->mcu_valid, *this->apt_valid,
100    *this->am_valid);
100 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



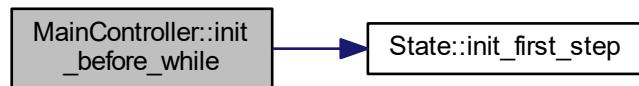
5.10.3.3 init_before_while()

```
void MainController::init_before_while ( )
```

Definition at line 4 of file main_controller.cpp.

```
5 {
6     this->state->init_first_step();
7 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.3.4 init_state()

```
void MainController::init_state ( )
```

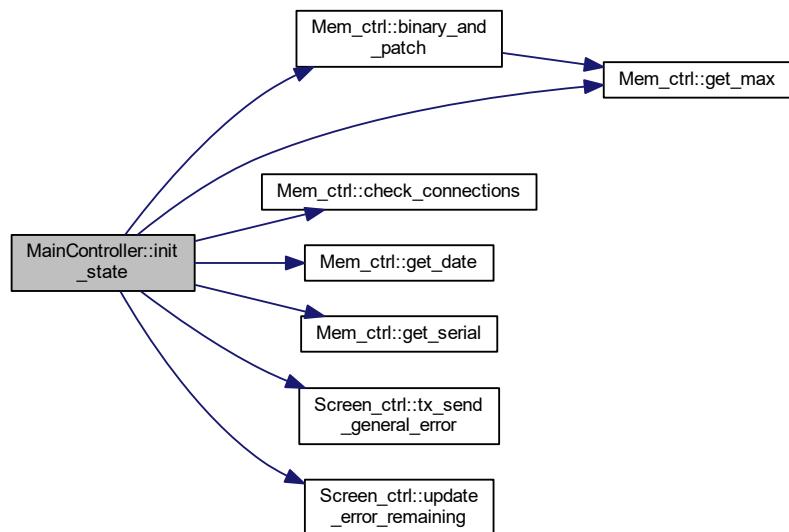
Definition at line 9 of file main_controller.cpp.

```
10 {
11     this->state->next = Init;
12     this->mem->check_connections();
13     this->debug->update();
14     if (this->state->bypass.value == 3)
15     {
16         this->state->next = Idle;
17     }
18     else if (this->state->bypass.value == 4)
19     {
20         this->state->next = QA;
21     }
22     else if (this->state->bypass.value == 8)
23     {
24         // In this state we check connections and connect apt and then pass uart state
25         this->state->next = UartDebug;
26     }
27     else
28     {
```

```

29     if (*this->mcu_valid && *this->apt_valid && *this->am_valid)
30     {
31         this->mem->get_serial();
32         this->mem->get_max();
33         this->mem->get_date();
34         this->mem->binary_and_patch();
35     }
36     else if (!*this->mcu_valid)
37     {
38         this->screen->tx_send_general_error(404);
39     }
40     else if (!*this->apt_valid)
41     {
42         this->screen->tx_send_general_error(405);
43     }
44     else if (!*this->am_valid)
45     {
46         this->screen->tx_send_general_error(406);
47     }
48
49     // Init step 2.
50     if(*this->am_sn == 0xffffffff || *this->am_sn == 0x00000000 || *this->max_count == 0xffffffff || *this->max_count == 0)
51     {
52         this->screen->tx_send_general_error(501);
53     }
54     else if(*this->pulses >= *this->max_count)
55     {
56         this->screen->update_error_remaining(0);
57     }
58     else
59     {
60         this->state->next = Idle;
61     }
62 }
63 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.3.5 need_to_stall()

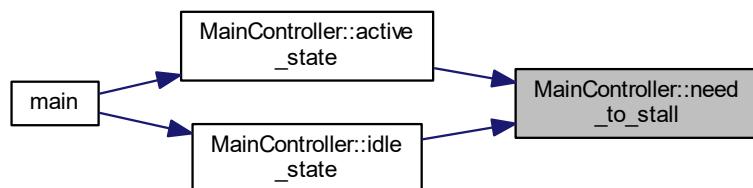
```
bool MainController::need_to_stall ( )
```

Definition at line 253 of file main_controller.cpp.

```

254 {
255     if (this->motor_running)
256     {
257         if (*this->pulses > *this->max_count)
258         {
259             return true;
260         }
261         if (!*this->pressure_ok)
262         {
263             return true;
264         }
265         if (!*this->am_valid)
266         {
267             return true;
268         }
269         if (this->pulses_on_screen == 0 && !this->is_counting_up)
270         {
271             this->is_counting_up = true;
272             return true;
273         }
274     }
275     else
276     {
277         return false;
278     }
279     return false;
280 }
```

Here is the caller graph for this function:



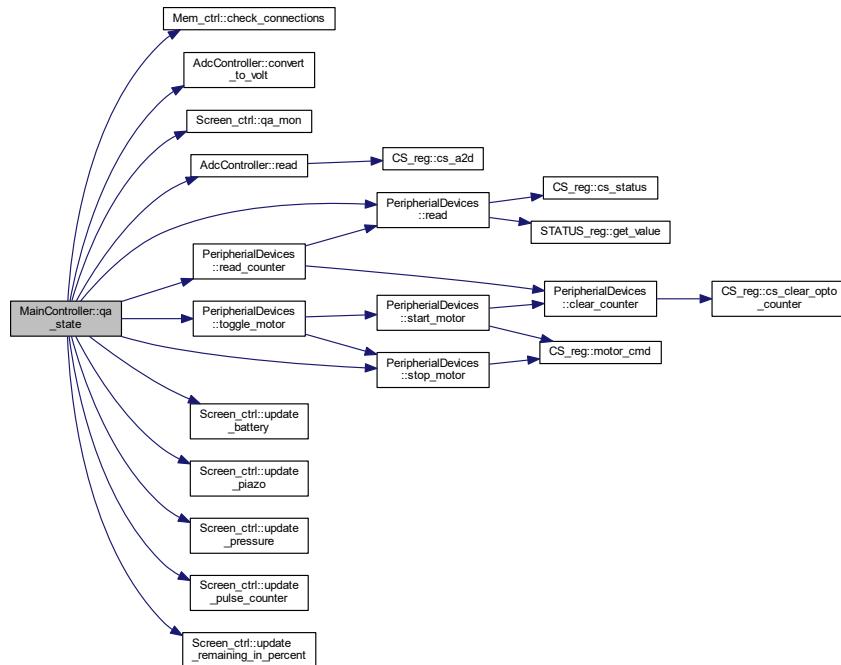
5.10.3.6 qa_state()

```
void MainController::qa_state ( )
```

Definition at line 152 of file main_controller.cpp.

```
153 {
154     this->adc->read();
155     this->adc->convert_to_volt();
156     this->device->read();
157     if (this->status->bits.FIRE_STATUS)
158     {
159         this->device->toggle_motor();
160     }
161
162     volatile uint32_t added_pulses = this->device->read_counter();
163     if (added_pulses > 0)
164     {
165         this->mem->AM_MEM->pulses += added_pulses;           // add a function to spi write and increment
166         this->mem->AM_MEM->cursor += added_pulses * 8;
167         if (this->is_counting_up)
168         {
169             this->pulses_on_screen += added_pulses;
170         }
171         else
172         {
173             if (this->pulses_on_screen > added_pulses)
174             {
175                 this->pulses_on_screen = this->pulses_on_screen - added_pulses;
176             }
177             else
178             {
179                 this->pulses_on_screen = 0;
180                 this->device->stop_motor();
181             }
182         }
183     }
184     this->screen->update_pulse_counter(*this->pulses);
185     this->mem->check_connections();
186     this->screen->update_battery(*this->battery, *this->battery_percent);
187     this->screen->update_pressure(*this->pressure);
188     this->screen->update_remaining_in_percent(10000, *this->pulses); //default is 10000 for initial run
189     this->screen->update_piazo(*this->motor_running);
190     // special to QA:
191     this->screen->qa_mon(*this->mcu_valid, *this->apt_valid, *this->am_valid);
192     this->debug->update(*this->motor_running, *this->pressure_ok, *this->mcu_valid, *this->apt_valid,
193     *this->am_valid);
```

Here is the call graph for this function:



Here is the caller graph for this function:



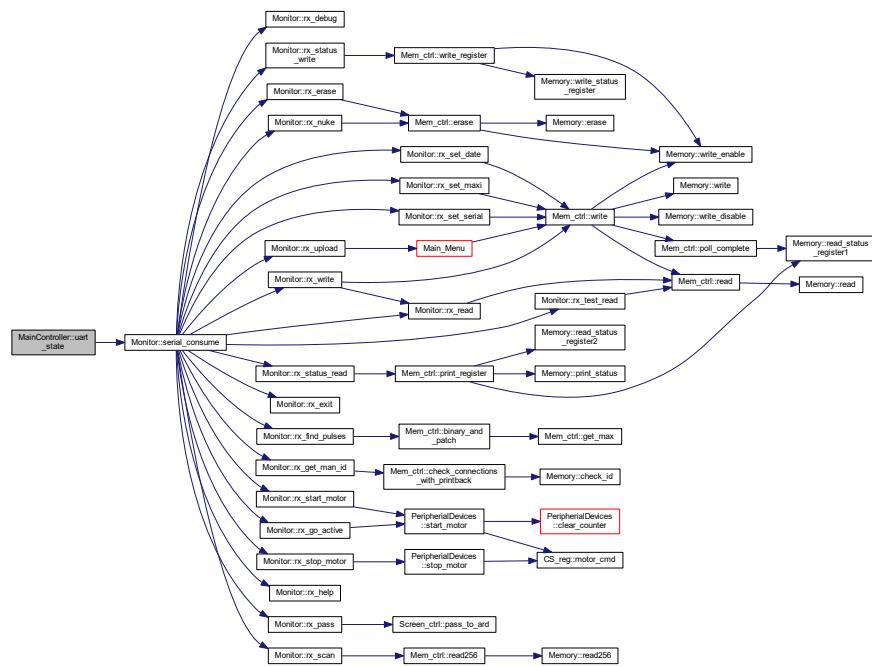
5.10.3.7 uart_state()

```
void MainController::uart_state (
    void )
```

Definition at line 239 of file main_controller.cpp.

```
240 {
241     Monitor monitor = Monitor(this);
242     while (this->state->current==UartDebug)
243     {
244         monitor.serial_consume();
245     }
246 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



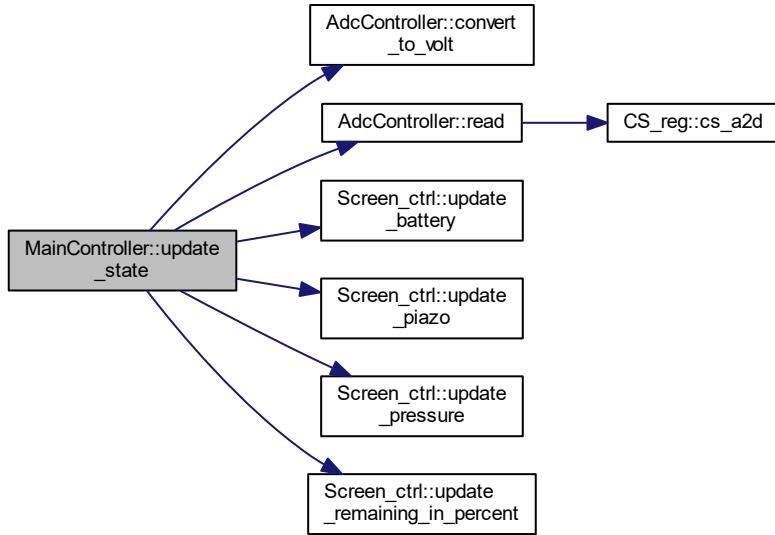
5.10.3.8 update_state()

```
void MainController::update_state ( )
```

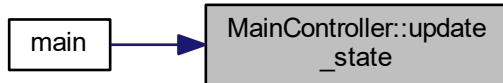
Definition at line 195 of file main_controller.cpp.

```
196 {
197     this->adc->read();
198     this->adc->convert_to_volt();
199     this->screen->update_battery(*this->battery, *this->battery_percent);
200     this->screen->update_pressure(*this->pressure);
201     this->screen->update_remaining_in_percent(*this->max_count, *this->pulses);
202     this->screen->update_piazo(*this->motor_running);
203 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.4 Member Data Documentation

5.10.4.1 adc

```
AdcController* MainController::adc
```

Definition at line 17 of file `main_controller.h`.

5.10.4.2 am_sn

```
uint32_t* MainController::am_sn
```

Definition at line 32 of file main_controller.h.

5.10.4.3 am_valid

```
bool* MainController::am_valid
```

Definition at line 23 of file main_controller.h.

5.10.4.4 apt_sn

```
uint32_t* MainController::apt_sn
```

Definition at line 31 of file main_controller.h.

5.10.4.5 apt_valid

```
bool* MainController::apt_valid
```

Definition at line 22 of file main_controller.h.

5.10.4.6 battery

```
int16_t* MainController::battery
```

Definition at line 24 of file main_controller.h.

5.10.4.7 battery_percent

```
uint32_t* MainController::battery_percent
```

Definition at line 25 of file main_controller.h.

5.10.4.8 cursor

```
uint32_t* MainController::cursor
```

Definition at line 33 of file main_controller.h.

5.10.4.9 debug

```
DebugHelper* MainController::debug
```

Definition at line 12 of file main_controller.h.

5.10.4.10 device

```
PeripheralDevices* MainController::device
```

Definition at line 14 of file main_controller.h.

5.10.4.11 is_counting_up

```
bool MainController::is_counting_up
```

Definition at line 34 of file main_controller.h.

5.10.4.12 max_count

```
uint32_t* MainController::max_count
```

Definition at line 19 of file main_controller.h.

5.10.4.13 mcu_sn

```
uint32_t* MainController::mcu_sn
```

Definition at line 30 of file main_controller.h.

5.10.4.14 mcu_valid

```
bool* MainController::mcu_valid
```

Definition at line 21 of file main_controller.h.

5.10.4.15 mem

```
Mem_ctrl* MainController::mem
```

Definition at line 13 of file main_controller.h.

5.10.4.16 motor_running

```
bool* MainController::motor_running
```

Definition at line 27 of file main_controller.h.

5.10.4.17 pressure

```
int16_t* MainController::pressure
```

Definition at line 26 of file main_controller.h.

5.10.4.18 pressure_ok

```
bool* MainController::pressure_ok
```

Definition at line 28 of file main_controller.h.

5.10.4.19 pulses

```
uint32_t* MainController::pulses
```

Definition at line 18 of file main_controller.h.

5.10.4.20 pulses_on_screen

```
uint32_t MainController::pulses_on_screen
```

Definition at line 20 of file main_controller.h.

5.10.4.21 screen

```
Screen_ctrl* MainController::screen
```

Definition at line 16 of file main_controller.h.

5.10.4.22 state

```
State* MainController::state
```

Definition at line 15 of file main_controller.h.

5.10.4.23 status

```
byte8_t_reg_status* MainController::status
```

Definition at line 29 of file main_controller.h.

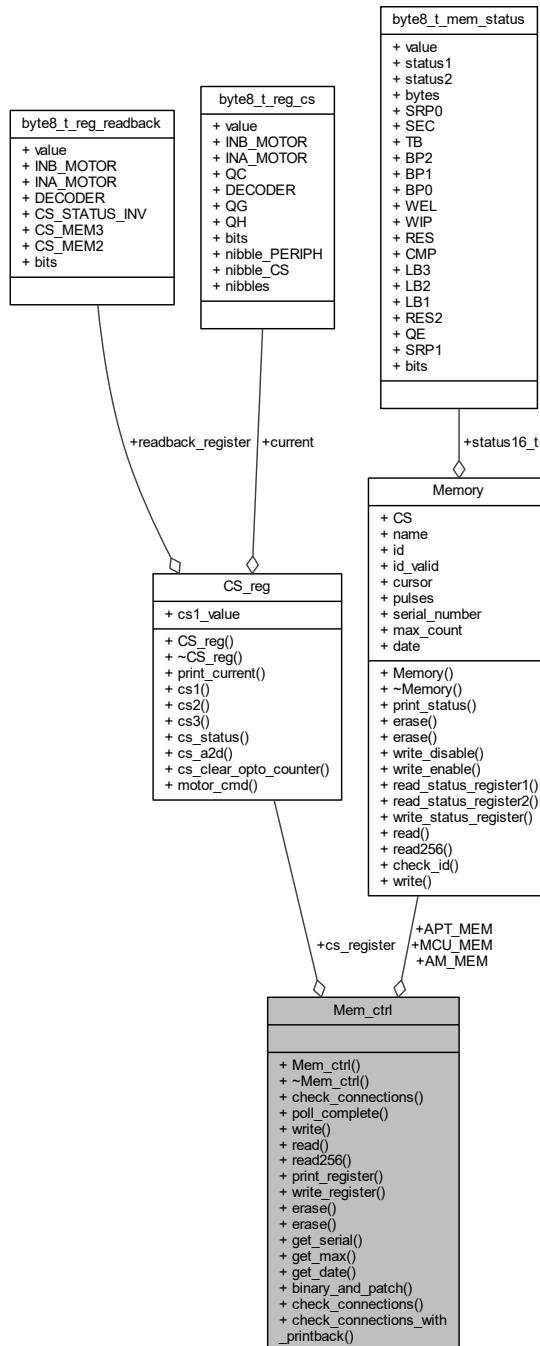
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[main_controller.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[main_controller.cpp](#)

5.11 Mem_ctrl Class Reference

```
#include <mem_controller.h>
```

Collaboration diagram for Mem_ctrl:



Public Member Functions

- `Mem_ctrl (Memory *, Memory *, Memory *, CS_reg *)`

- `~Mem_ctrl ()`
- `void check_connections (void)`
- `HAL_StatusTypeDef poll_complete (uint8_t cs_num, uint32_t timeout)`
- `HAL_StatusTypeDef write (uint8_t cs_num, uint32_t write_address, uint32_t write_value)`
- `uint32_t read (uint8_t, uint32_t)`
- `void read256 (uint8_t, uint32_t)`
- `void print_register (uint8_t)`
- `void write_register (uint8_t, uint16_t)`
- `void erase (uint8_t cs_num, uint32_t address)`
- `void erase (uint8_t cs_num)`
- `void get_serial (void)`
- `void get_max (void)`
- `void get_date (void)`
- `uint32_t binary_and_patch (void)`
- `bool check_connections (uint8_t)`
- `bool check_connections_with_printback (uint8_t)`

Public Attributes

- `Memory * MCU_MEM`
- `Memory * APT_MEM`
- `Memory * AM_MEM`
- `CS_reg * cs_register`

5.11.1 Detailed Description

Definition at line 5 of file mem_controller.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Mem_ctrl()

```
Mem_ctrl::Mem_ctrl (
    Memory * AM_MEM,
    Memory * APT_MEM,
    Memory * MCU_MEM,
    CS_reg * cs_register )
```

Definition at line 8 of file mem_controller.cpp.

```
9 {
10     this->AM_MEM = AM_MEM;
11     this->APT_MEM = APT_MEM;
12     this->MCU_MEM = MCU_MEM;
13     this->cs_register = cs_register;
14 }
```

5.11.2.2 ~Mem_ctrl()

```
Mem_ctrl::~Mem_ctrl ( )
```

Definition at line 15 of file mem_controller.cpp.

```
16 {  
17  
18 }
```

5.11.3 Member Function Documentation

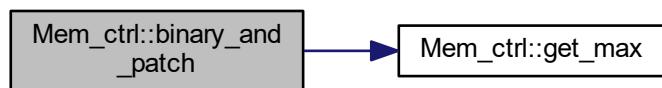
5.11.3.1 binary_and_patch()

```
uint32_t Mem_ctrl::binary_and_patch ( void )
```

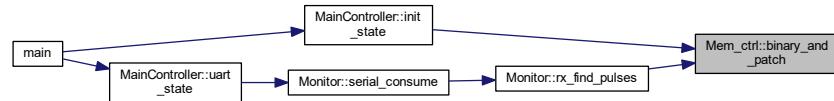
Definition at line 204 of file mem_controller.cpp.

```
205 {  
206     this->get_max(3);  
207     uint32_t address_pulse = 0;  
208     while (true)  
209     {  
210         address_pulse = this->get_ap_offset();  
211         if (address_pulse < this->AM_MEM->max_count * 8 - MAX_HOLE_SIZE * 4)  
212         {  
213             if (this->check_for_holes(address_pulse))  
214             {  
215                 this->patch_mem(address_pulse);  
216             }  
217             else  
218             {  
219                 break;  
220             }  
221         }  
222         else  
223         {  
224             break;  
225         }  
226     }  
227     this->AM_MEM->cursor = address_pulse;  
228     if (this->AM_MEM->cursor != 0)  
229     {  
230         this->AM_MEM->pulses = (address_pulse - 4) / 8 + 1;  
231     }  
232     else  
233     {  
234         this->AM_MEM->pulses = 0;  
235     }  
236     return address_pulse;  
237 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.2 check_connections() [1/2]

```
bool Mem_ctrl::check_connections (
    uint8_t cs_num )
```

Definition at line 56 of file mem_controller.cpp.

```
57 {
58     Memory* current_MEMORY = this->chip_select_pull(cs_num, LOW);      // PULL CS LOW so we start to send data
59     current_MEMORY->check_id();
60     this->chip_select_pull(cs_num, HIGH);
61     return current_MEMORY->id_valid;
62 }
```

Here is the call graph for this function:



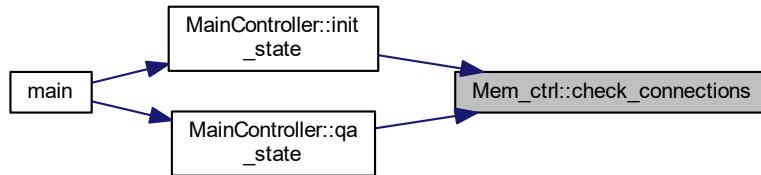
5.11.3.3 check_connections() [2/2]

```
void Mem_ctrl::check_connections (
    void )
```

Definition at line 78 of file mem_controller.cpp.

```
79 {
80     for (uint8_t cs = 1; cs < 4; cs++)
81     {
82         this->check_connections(cs);
83         HAL_Delay(1);
84     }
85
86 }
```

Here is the caller graph for this function:



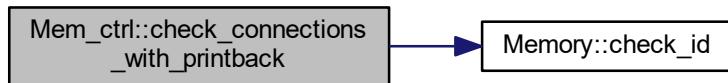
5.11.3.4 check_connections_with_printback()

```
bool Mem_ctrl::check_connections_with_printback (
    uint8_t cs_num )
```

Definition at line 64 of file mem_controller.cpp.

```
65 {
66     Memory* current_MEMORY = this->chip_select_pull(cs_num, LOW);      // PULL CS LOW so we start to send
67     data
68     current_MEMORY->check_id();
69     this->chip_select_pull(cs_num, HIGH);
70     char id_str[40];
71     sprintf(id_str, "\r\nCS[%d]: get id: %08X-%08X-%08X\r\n", cs_num,
72             current_MEMORY->id[0], \
73             current_MEMORY->id[1], \
74             current_MEMORY->id[2]);
75     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)id_str, strlen(id_str), 10);
76     return current_MEMORY->id_valid;
77 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



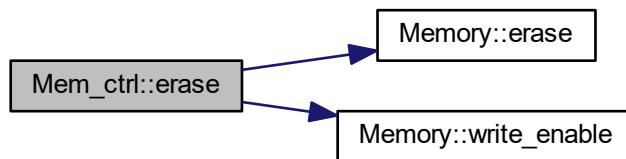
5.11.3.5 `erase()` [1/2]

```
void Mem_ctrl::erase (
    uint8_t cs_num )
```

Definition at line 366 of file mem_controller.cpp.

```
367 {
368     Memory* current_mem = this->chip_select_pull(cs_num, LOW);
369     current_mem->write_enable();
370     this->chip_select_pull(cs_num, HIGH);
371     HAL_Delay(1);
372     this->chip_select_pull(cs_num, LOW);
373     current_mem->erase();
374     this->chip_select_pull(cs_num, HIGH);
375 }
```

Here is the call graph for this function:



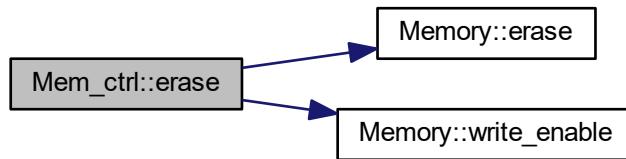
5.11.3.6 `erase()` [2/2]

```
void Mem_ctrl::erase (
    uint8_t cs_num,
    uint32_t address )
```

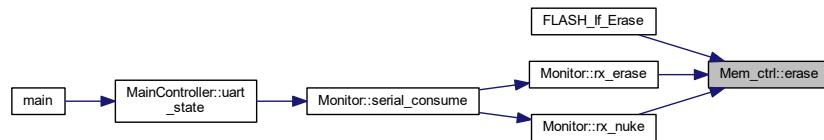
Definition at line 355 of file mem_controller.cpp.

```
356 {
357     Memory* current_mem = this->chip_select_pull(cs_num, LOW);
358     current_mem->write_enable();
359     this->chip_select_pull(cs_num, HIGH);
360     HAL_Delay(1);
361     this->chip_select_pull(cs_num, LOW);
362     current_mem->erase(address);
363     this->chip_select_pull(cs_num, HIGH);
364 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



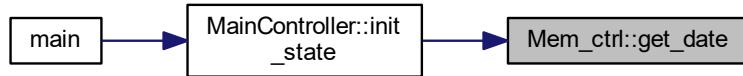
5.11.3.7 get_date()

```
void Mem_ctrl::get_date (
    void )
```

Definition at line 326 of file mem_controller.cpp.

```
327 {
328     this->get_date(3);
329     HAL_Delay(1);
330 }
```

Here is the caller graph for this function:



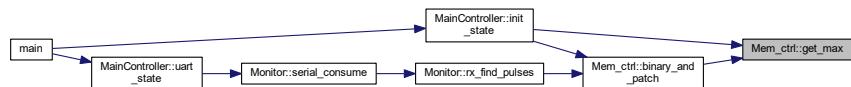
5.11.3.8 get_max()

```
void Mem_ctrl::get_max (
    void )
```

Definition at line 320 of file mem_controller.cpp.

```
321 {
322     this->get_max(3);
323     HAL_Delay(1);
324 }
```

Here is the caller graph for this function:



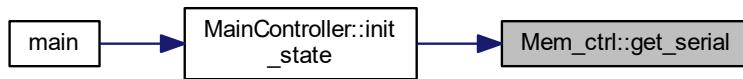
5.11.3.9 get_serial()

```
void Mem_ctrl::get_serial (
    void )
```

Definition at line 312 of file mem_controller.cpp.

```
313 {
314     for (uint8_t cs = 1; cs < 4; cs++)
315     {
316         this->get_serial(cs);
317         HAL_Delay(1);
318     }
319 }
```

Here is the caller graph for this function:



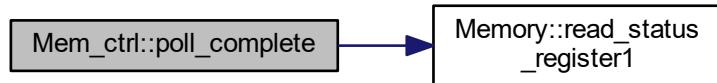
5.11.3.10 poll_complete()

```
HAL_StatusTypeDef Mem_ctrl::poll_complete (
    uint8_t cs_num,
    uint32_t timeout )
```

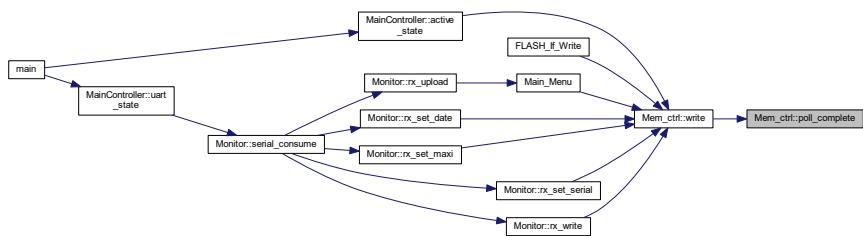
Definition at line 112 of file mem_controller.cpp.

```
113 {
114     Memory* current_MEMORY = this->chip_select_pull(cs_num, LOW);           // PULL CS LOW so we start to send
115     data
116     volatile uint32_t time_start = HAL_GetTick();
117
118     if (current_MEMORY->id_valid)
119     {
120         current_MEMORY->read_status_register1();
121         this->chip_select_pull(cs_num, HIGH);
122         HAL_Delay(1);
123
124         while ((current_MEMORY->status16_t.bits.WIP) == 1) // while busy
125         {
126             this->chip_select_pull(cs_num, LOW);           // PULL CS LOW so we start to send data
127             current_MEMORY->read_status_register1();
128             this->chip_select_pull(cs_num, HIGH);
129             HAL_Delay(1);
130             if (HAL_GetTick() - time_start > timeout)
131             {
132                 return HAL_TIMEOUT;
133             }
134         }
135     }
136     else
137     {
138         uint8_t temp[40];
139         sprintf((char*)temp, "SPI action could not complete, %s mem not connected
140 \r\n", current_MEMORY->name);
141         HAL_UART_Transmit(SERIAL_PC, temp, strlen((char*)temp), 100);
142         return HAL_ERROR;
143     }
144 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



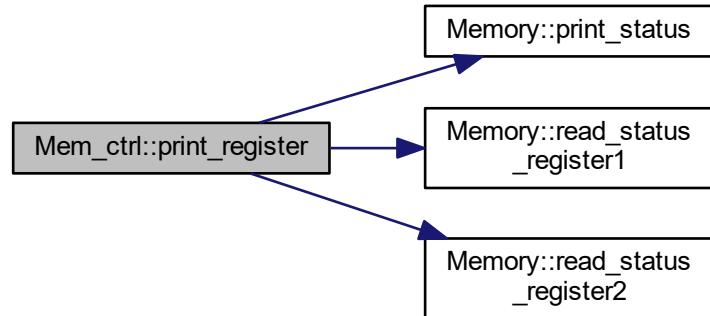
5.11.3.11 print_register()

```
void Mem_ctrl::print_register (
    uint8_t cs_num )
```

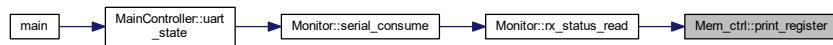
Definition at line 332 of file mem_controller.cpp.

```
333 {
334     Memory* current_mem = this->chip_select_pull(cs_num, LOW);
335     current_mem->read_status_register1();
336     this->chip_select_pull(cs_num, HIGH);
337     HAL_Delay(1);
338     this->chip_select_pull(cs_num, LOW);
339     current_mem->read_status_register2();
340     this->chip_select_pull(cs_num, HIGH);
341     current_mem->print_status();
342 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.12 `read()`

```

uint32_t Mem_ctrl::read (
    uint8_t cs_num,
    uint32_t read_address )
  
```

Definition at line 40 of file `mem_controller.cpp`.

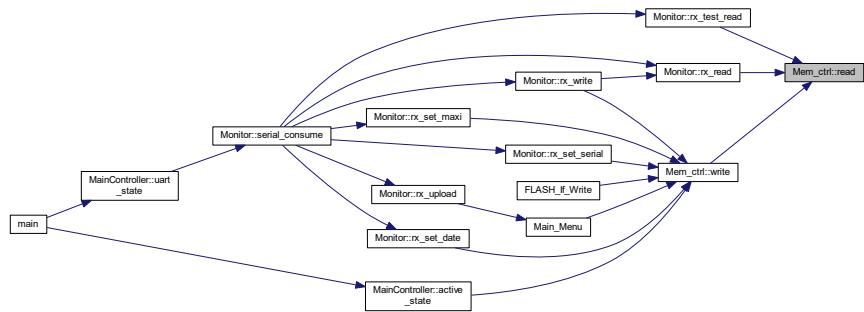
```

41 {
42     Memory* current_MEMORY = this->chip_select_pull(cs_num, LOW); // PULL CS LOW so we start to send data
43     uint32_t read_value = current_MEMORY->read(read_address);
44     this->chip_select_pull(cs_num, HIGH);
45     return read_value;
46 }
  
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.3.13 read256()

```
void Mem_ctrl::read256 (
    uint8_t cs_num,
    uint32_t read_address )
```

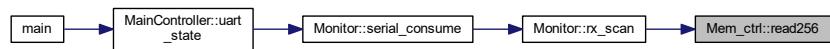
Definition at line 48 of file mem_controller.cpp.

```
49 {
50     Memory* current_MEMORY = this->chip_select_pull(cs_num, LOW); // PULL CS LOW so we start to send data
51     current_MEMORY->read256(read_address);
52     this->chip_select_pull(cs_num, HIGH);
53 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



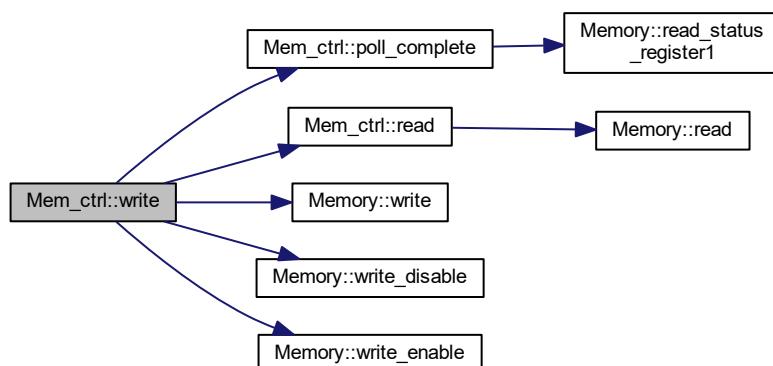
5.11.3.14 write()

```
HAL_StatusTypeDef Mem_ctrl::write (
    uint8_t cs_num,
    uint32_t write_address,
    uint32_t write_value )
```

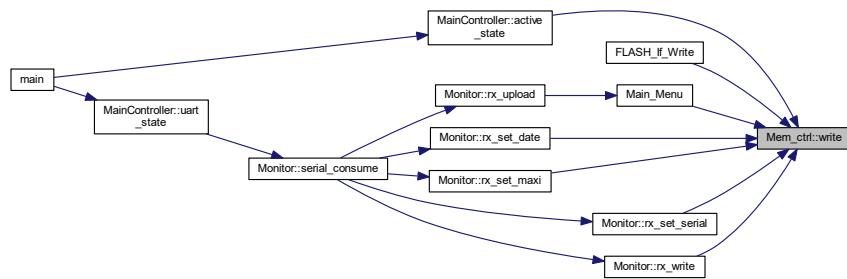
Definition at line 146 of file mem_controller.cpp.

```
147 {
148     HAL_StatusTypeDef status = HAL_OK;
149     // -----
150     // write start : Counter
151     // counter value
152     //spi_erase_4_kb_from_ff(ChipSelectNumber, write_address);
153     Memory* current_MEMORY = this->chip_select_pull(cs_num, LOW);
154
155     if (current_MEMORY->write_enable() != HAL_OK)
156     {
157         this->chip_select_pull(cs_num, HIGH);
158         return HAL_ERROR;
159     }
160     this->chip_select_pull(cs_num, HIGH);
161     HAL_Delay(1);
162
163     this->chip_select_pull(cs_num, LOW);
164     current_MEMORY->write(write_address, write_value);
165     this->chip_select_pull(cs_num, HIGH);
166     HAL_Delay(1);
167
168     status = this->poll_complete(cs_num, 30);
169     if (status != HAL_OK)
170     {
171         return HAL_ERROR;
172     }
173     HAL_Delay(1);
174
175     this->chip_select_pull(cs_num, LOW);
176     if (current_MEMORY->write_disable() != HAL_OK)
177     {
178         this->chip_select_pull(cs_num, HIGH);
179         return HAL_ERROR;
180     }
181     this->chip_select_pull(cs_num, HIGH);
182     HAL_Delay(1);
183
184     volatile uint32_t read_back = this->read(cs_num, write_address);
185     if (read_back != write_value)
186     {
187         return HAL_ERROR;
188     }
189     return HAL_OK;
190 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



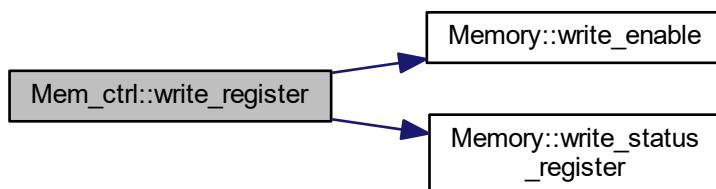
5.11.3.15 write_register()

```
void Mem_ctrl::write_register (
    uint8_t cs_num,
    uint16_t value )
```

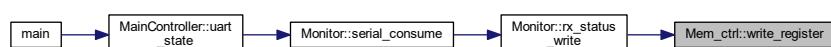
Definition at line 344 of file mem_controller.cpp.

```
345 {
346     Memory* current_mem = this->chip_select_pull(cs_num, LOW);
347     current_mem->write_enable();
348     this->chip_select_pull(cs_num, HIGH);
349     HAL_Delay(1);
350     this->chip_select_pull(cs_num, LOW);
351     current_mem->write_status_register(value);
352     this->chip_select_pull(cs_num, HIGH);
353 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.11.4 Member Data Documentation

5.11.4.1 AM_MEM

`Memory* Mem_ctrl::AM_MEM`

Definition at line 10 of file mem_controller.h.

5.11.4.2 APT_MEM

`Memory* Mem_ctrl::APT_MEM`

Definition at line 9 of file mem_controller.h.

5.11.4.3 cs_register

`CS_Reg* Mem_ctrl::cs_register`

Definition at line 11 of file mem_controller.h.

5.11.4.4 MCU_MEM

`Memory* Mem_ctrl::MCU_MEM`

Definition at line 8 of file mem_controller.h.

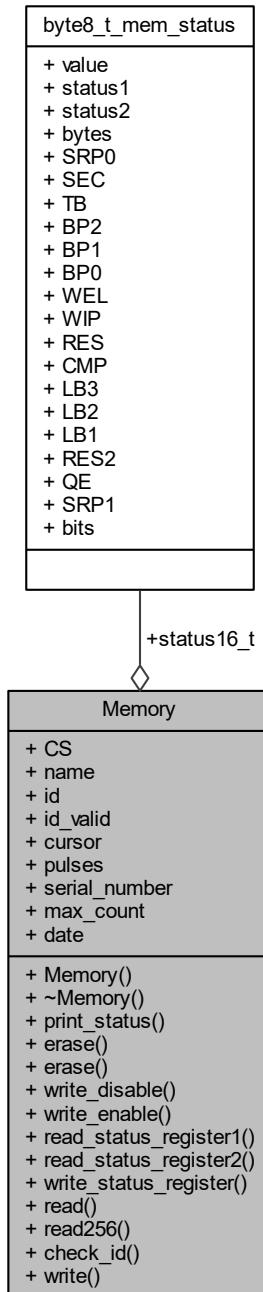
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[mem_controller.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[mem_controller.cpp](#)

5.12 Memory Class Reference

```
#include <mem.h>
```

Collaboration diagram for Memory:



Public Member Functions

- [Memory \(uint8_t\)](#)

- `~Memory ()`
- `void print_status (void)`
- `HAL_StatusTypeDef erase (uint32_t erase_address)`
- `HAL_StatusTypeDef erase (void)`
- `HAL_StatusTypeDef write_disable (void)`
- `HAL_StatusTypeDef write_enable (void)`
- `void read_status_register1 (void)`
- `void read_status_register2 (void)`
- `void write_status_register (uint16_t write_now)`
- `uint32_t read (uint32_t read_address)`
- `void read256 (uint32_t read_address)`
- `void check_id (void)`
- `HAL_StatusTypeDef write (uint32_t write_address, uint32_t write_value)`

Public Attributes

- `uint8_t CS`
- `char name [5]`
- `uint8_t id [3]`
- `bool id_valid`
- `byte8_t_mem_status status16_t`
- `uint32_t cursor`
- `uint32_t pulses`
- `uint32_t serial_number`
- `uint32_t max_count`
- `uint32_t date`

5.12.1 Detailed Description

Definition at line 41 of file mem.h.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Memory()

```
Memory::Memory (
    uint8_t cs_num )
```

Definition at line 8 of file mem.cpp.

```
9 {
10     this->CS = cs_num; // 1, 2, 3
11     if(this->CS == 1)
12     {
13         sprintf((char*)this->name, "MCU");
14     }
15     else if(this->CS == 2)
16     {
17         sprintf((char*)this->name, "APT");
18     }
19     else if(this->CS == 3)
20     {
21         sprintf((char*)this->name, "AM");
22     }
```

```

23     this->id[0] = 0;    // valid id would be 0x1f-85-01 or something
24     this->id[1] = 0;    // valid id would be 0x1f-85-01 or something
25     this->id[2] = 0;    // valid id would be 0x1f-85-01 or something
26     this->id_valid = false; // true if okay
27     this->status16_t.value = 0;
28     this->cursor = 0;    // pointer to read memory
29     this->pulses = 0;
30     this->serial_number = 0;
31     this->max_count = 0;
32     this->date = 0;
33 }

```

5.12.2.2 ~Memory()

Memory::~Memory ()

Definition at line 35 of file mem.cpp.

```

36 {
37
38 }

```

5.12.3 Member Function Documentation

5.12.3.1 check_id()

```
void Memory::check_id (
    void )
```

Definition at line 132 of file mem.cpp.

```

133 {
134     this->get_id();
135     if ((this->id[0] == 0x1f)&&(this->id[1] == 0x85)&&(this->id[2] == 0x01))
136     {
137         this->id_valid = true;
138     }
139     else
140     {
141         this->id_valid = false;
142     }
143 }
```

Here is the caller graph for this function:



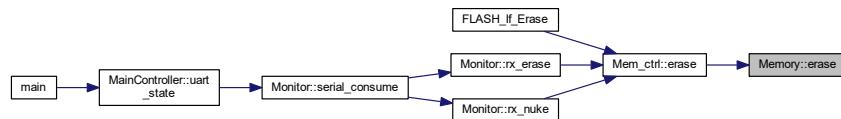
5.12.3.2 `erase()` [1/2]

```
HAL_StatusTypeDef Memory::erase (
    uint32_t erase_address )
```

Definition at line 109 of file mem.cpp.

```
110 {
111     HAL_StatusTypeDef status = HAL_OK;
112     uint8_t erase_command[1];
113     erase_command[0] = BLOCK_ERASE_4K;      // erase opcodes: 4Kbytes: 0x20, 32Kbytes: 0x52, 64Kbytes:
114     0xD8
115     status = HAL_SPI_Transmit(&hspil, erase_command, 1, 500);
116     if(status != HAL_OK)
117     {
118         return status;
119     }
120     // Address :
121     uint8_t address[3];
122     this->translate_int_to_address(erase_address, address);
123     return HAL_SPI_Transmit(&hspil, address, 3, 500);
124 }
```

Here is the caller graph for this function:



5.12.3.3 `erase()` [2/2]

```
HAL_StatusTypeDef Memory::erase (
    void )
```

Definition at line 125 of file mem.cpp.

```
126 {
127     uint8_t erase_command[1];
128     erase_command[0] = ERASE_CHIP2;      // erase the entire chip
129     return HAL_SPI_Transmit(&hspil, erase_command, 1, 2500) ;
130 }
```

5.12.3.4 `print_status()`

```
void Memory::print_status (
    void )
```

Definition at line 40 of file mem.cpp.

```
41 {
42
43     uint8_t message[128];
44     sprintf((char*)message,
45             "cs[%d] read %08X\r\n\"\
46             "SRP0 [%d]\r\n"\
47             "SEC  [%d]\r\n"\
48             "TB   [%d]\r\n"\
49             "BP2  [%d]\r\n\"
```

```

50          "BP1  [%d]\r\n"\
51          "BP0  [%d]\r\n"\
52          "WEL  [%d]\r\n"\
53          "WIP  [%d]\r\n"\
54          "\r\n"\
55          "RES  [%d]\r\n"\
56          "CMP  [%d]\r\n"\
57          "LB3  [%d]\r\n"\
58          "LB2  [%d]\r\n"\
59          "LB1  [%d]\r\n"\
60          "RES  [%d]\r\n"\
61          "QE   [%d]\r\n"\
62          "SRP1 [%d]\r\n",
63      this->CS,
64      this->status16_t.value,
65      this->status16_t.bits.SRP0,
66      this->status16_t.bits.SEC,
67      this->status16_t.bits.TB,
68      this->status16_t.bits.BP2,
69      this->status16_t.bits.BP1,
70      this->status16_t.bits.BP0,
71      this->status16_t.bits.WEL,
72      this->status16_t.bits.WIP,
73      this->status16_t.bits.RES,
74      this->status16_t.bits.CMP,
75      this->status16_t.bits.LB3,
76      this->status16_t.bits.LB2,
77      this->status16_t.bits.LB1,
78      this->status16_t.bits.RES2,
79      this->status16_t.bits.QE,
80      this->status16_t.bits.SRP1);
81     HAL_UART_Transmit(SERIAL_PC, message, strlen((char*)message), 100);
82 }

```

Here is the caller graph for this function:



5.12.3.5 read()

```

uint32_t Memory::read (
    uint32_t read_address )

```

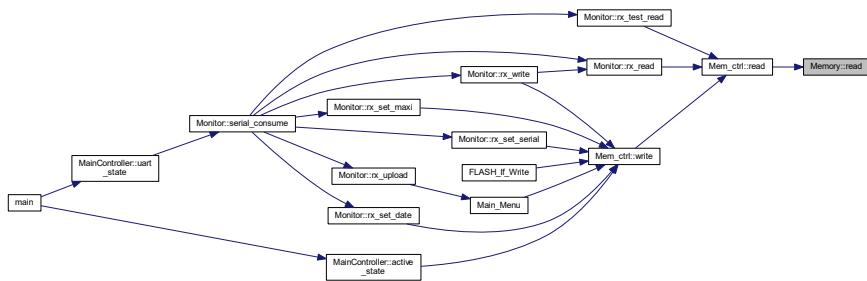
Definition at line 223 of file mem.cpp.

```

224 {
225
226     uint8_t spi_read_command[1];
227     spi_read_command[0] = READ_ARRAY1;
228     /*Read an array
229     *0x0B needs an additional dummy byte to transmit data
230     *difference is 0x03 can be used only at lower speeds
231     *0x0B can be used at any speed
232     *ONLY 0x0B needs a dummy byte
233 */
234     HAL_SPI_Transmit(&hspil, spi_read_command, 1, 500);
235
236     uint8_t address[4];
237     uint8_t spi_read_data[4];
238     this->translate_int_to_address(read_address, address);
239     address[3] = 0xFF; //doesn't matter
240
241     HAL_SPI_Transmit(&hspil, address, 4, 500);
242     HAL_SPI_Receive(&hspil, spi_read_data, 4, 1000);
243
244     return this->translate_address_to_int(spi_read_data);
245 }

```

Here is the caller graph for this function:



5.12.3.6 read256()

```
void Memory::read256 (
```

Definition at line 247 of file mem.cpp.

```

248 {
249
250     uint8_t spi_read_command[1];
251     spi_read_command[0] = READ_ARRAY1;
252     /*Read an array
253     *0x0B needs an additional dummy byte to transmit data
254     *difference is 0x03 can be used only at lower speeds
255     *0x0B can be used at any speed
256     *ONLY 0x0B needs a dummy byte
257     */
258     HAL_SPI_Transmit(&hsp1, spi_read_command, 1, 500);
259
260     uint8_t address[4];
261     uint8_t spi_read_data;
262     this->translate_int_to_address(read_address, address);
263     address[3] = 0xFF;      //doesn't matter
264
265     HAL_SPI_Transmit(&hsp1, address, 4, 5);
266     for (uint8_t i = 0; i < 255; i++)
267     {
268         HAL_SPI_Receive(&hsp1, &spi_read_data, 1, 1);
269         static uint8_t value[10];
270         if (i % 16 == 15)
271         {
272             sprintf((char*)value, "0x%02x\r\n", spi_read_data & 0xff);
273         }
274         else
275         {
276             sprintf((char*)value, "0x%02x ", spi_read_data & 0xff);
277         }
278         HAL_UART_Transmit(SERIAL_PC, value, strlen((char*)value), 1);
279     }
280 }
```

Here is the caller graph for this function:



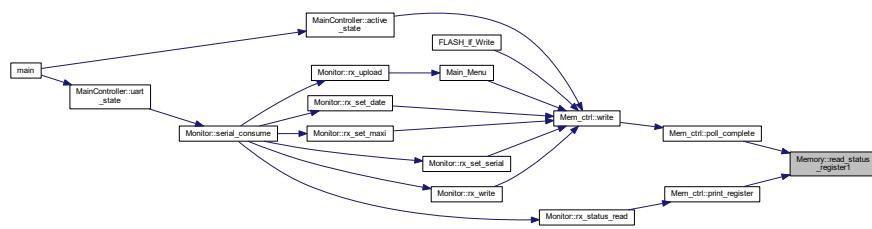
5.12.3.7 read_status_register1()

```
void Memory::read_status_register1 (
    void )
```

Definition at line 167 of file mem.cpp.

```
168 {
169     /*
170      7 SRP0 Status Register Protection bit-0 R/W See Table 10-3 on Status Register Protection
171      6 SEC Block Protection R/W See Table 8-1 and 8-2 on Non-Volatile Protection
172      5 TB Top or Bottom Protection R/W See Table 8-1 and 8-2 on Non-Volatile Protection
173      4 BP2 Block Protection bit-2 R/W See Table 8-1 and 8-2 on Non-Volatile Protection
174      3 BP1 Block Protection bit-1 R/W See Table 8-1 and 8-2 on Non-Volatile Protection
175      2 BPO Block Protection bit-0 R/W See Table 8-1 and 8-2 on Non-Volatile Protection
176      1 WEL Write Enable Latch Status R 0 Device is not Write Enabled (default)
177      0 RDY/BSY
178
179 */
180     uint8_t cmd[1];
181     cmd[0] = READ_STATUS_REGISTER1;
182
183     HAL_SPI_Transmit(&hsp1, cmd, 1, 1000);
184     HAL_SPI_Receive(&hsp1, &this->status16_t.bytes.status1, 1, 1000);
185 }
```

Here is the caller graph for this function:



5.12.3.8 read_status_register2()

```
void Memory::read_status_register2 (
    void )
```

Definition at line 187 of file mem.cpp.

```
188 {
189     /*
190      7 RES Reserve for future use R 0 Reserve for future use
191      6 CMP Complement Block Protection R/W 0 See table on Block Protection
192      5 LB3 Lock Security Register 3 R/W
193          0 Security Register page-3 is not locked (default)
194          1 Security Register page-3 cannot be erased/programmed
195      4 LB2 Lock Security Register 2 R/W
196          0 Security Register page-2 is not locked (default)
197          1 Security Register page-2 cannot be erased/programmed
198      3 LB1 Lock Security Register 1 R/W
199          0 Security Register page-1 is not locked (default)
200          1 Security Register page-1 cannot be erased/programmed
201      2 RES Reserved for future use R 0 Reserved for future use
202          1 QE Quad Enable R/W
203          0 HOLD and WP function normally (default)
204      1 HOLD and WP are I/O pins
205          0 SRP1 Status Register Protect bit-1 R/W See table on Status Register Protection
206
207 */
208     uint8_t cmd[1];
209     cmd[0] = READ_STATUS_REGISTER2;
210
211     HAL_SPI_Transmit(&hsp1, cmd, 1, 1000);
```

```
211     HAL_SPI_Receive(&hsp1, &this->status16_t.bytes.status2, 1, 1000);
212 }
```

Here is the caller graph for this function:



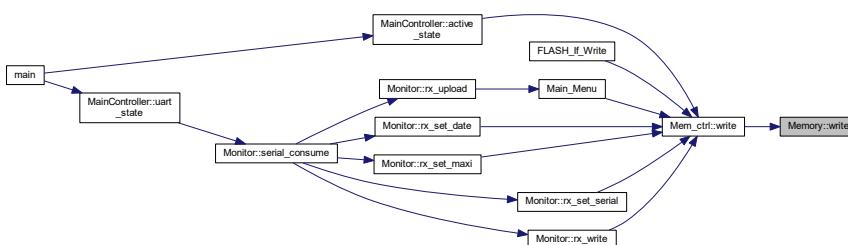
5.12.3.9 write()

```
HAL_StatusTypeDef Memory::write (
    uint32_t write_address,
    uint32_t write_value )
```

Definition at line 282 of file mem.cpp.

```
283 {
284     HAL_StatusTypeDef status = HAL_OK;
285
286     uint8_t spi_read_command[1];
287     spi_read_command[0] = BYTE_PROGRAM;
288     // Address start : where to write :
289     uint8_t address[4];
290     uint8_t counter_in_bytes[4];
291     volatile uint8_t complete_write[8];
292
293     address[3] = 0xFF;           //doesn't matter - is not sent
294     translate_int_to_address(write_address, address);
295     translate_int_to_arr(write_value, counter_in_bytes);
296     // Address end :
297
298     // after push address start to write data one by one ....
299     //      until length 256 total per page
300
301     // datas start from address start 00-0F-FFh 00-00-00h
302
303     complete_write[0] = spi_read_command[0];
304     complete_write[1] = address[0];
305     complete_write[2] = address[1];
306     complete_write[3] = address[2];
307     complete_write[4] = counter_in_bytes[0];
308     complete_write[5] = counter_in_bytes[1];
309     complete_write[6] = counter_in_bytes[2];
310     complete_write[7] = counter_in_bytes[3];
311     return HAL_SPI_Transmit(&hsp1, (uint8_t *)complete_write, 8, 500);
312 }
```

Here is the caller graph for this function:



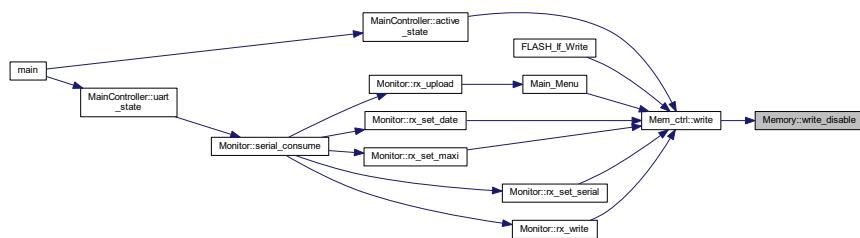
5.12.3.10 write_disable()

```
HAL_StatusTypeDef Memory::write_disable (
```

Definition at line 153 of file mem.cpp.

```
154 {
155     uint8_t spi_write_disable[1];
156     spi_write_disable[0] = WRITE_DISABLE;          // write enable
157     return HAL_SPI_Transmit(&hspil, spi_write_disable, 1, 500);
158 }
```

Here is the caller graph for this function:



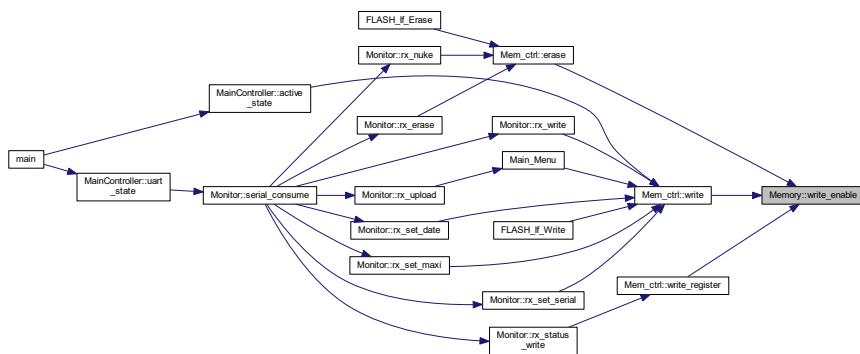
5.12.3.11 write_enable()

```
HAL_StatusTypeDef Memory::write_enable (
```

Definition at line 160 of file mem.cpp.

```
161 {
162     uint8_t spi_write_enable[1];
163     spi_write_enable[0] = WRITE_ENABLE;           // write enable
164     return HAL_SPI_Transmit(&hsp1, spi_write_enable, 1, 2);
165 }
```

Here is the caller graph for this function:



5.12.3.12 write_status_register()

```
void Memory::write_status_register (
    uint16_t write_now )
```

Definition at line 214 of file mem.cpp.

```
215 {
216     uint8_t cmd[3];
217     cmd[0] = WRITE_STATUS_REGISTER;
218     cmd[1] = (uint8_t)((write_now >> 8) & 0xff);
219     cmd[2] = (uint8_t)((write_now >> 0) & 0xff);
220     HAL_SPI_Transmit(&hspil, cmd, 3, 1000);
221 }
```

Here is the caller graph for this function:



5.12.4 Member Data Documentation

5.12.4.1 CS

```
uint8_t Memory::CS
```

Definition at line 44 of file mem.h.

5.12.4.2 cursor

```
uint32_t Memory::cursor
```

Definition at line 49 of file mem.h.

5.12.4.3 date

```
uint32_t Memory::date
```

Definition at line 53 of file mem.h.

5.12.4.4 **id**

```
uint8_t Memory::id[3]
```

Definition at line 46 of file mem.h.

5.12.4.5 **id_valid**

```
bool Memory::id_valid
```

Definition at line 47 of file mem.h.

5.12.4.6 **max_count**

```
uint32_t Memory::max_count
```

Definition at line 52 of file mem.h.

5.12.4.7 **name**

```
char Memory::name[5]
```

Definition at line 45 of file mem.h.

5.12.4.8 **pulses**

```
uint32_t Memory::pulses
```

Definition at line 50 of file mem.h.

5.12.4.9 **serial_number**

```
uint32_t Memory::serial_number
```

Definition at line 51 of file mem.h.

5.12.4.10 status16_t

```
byte8_t_mem_status Memory::status16_t
```

Definition at line 48 of file mem.h.

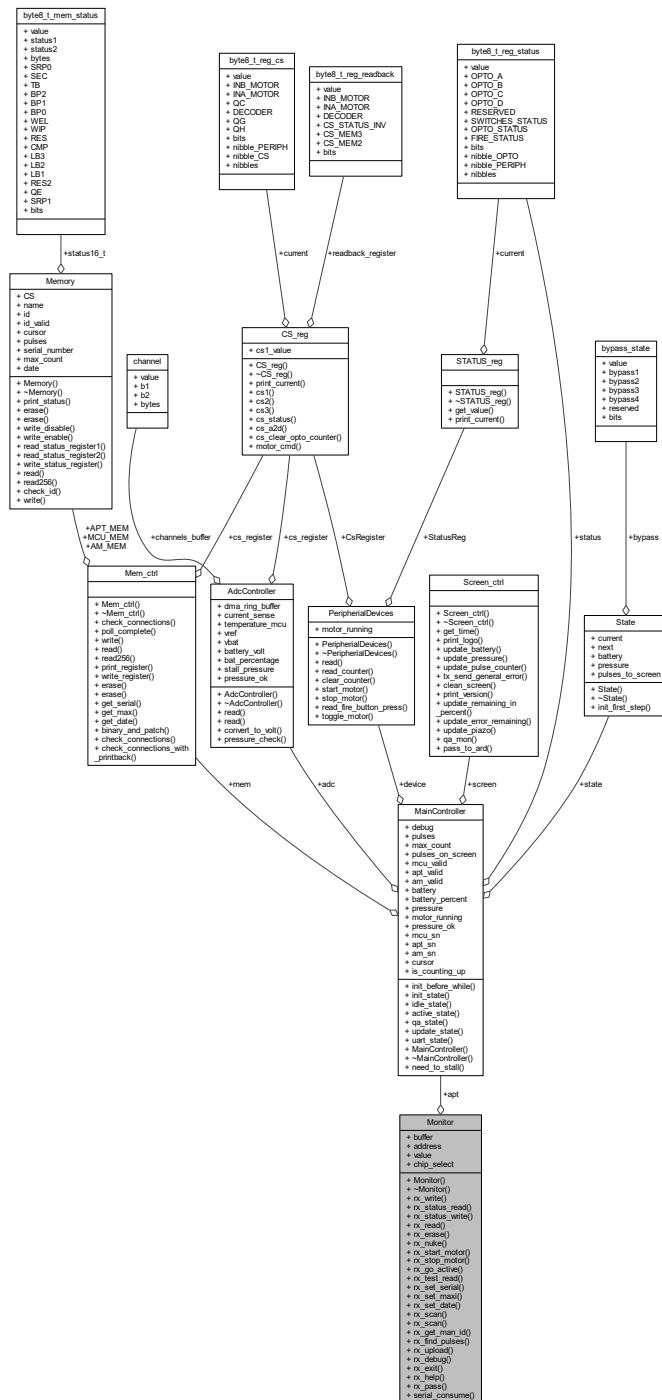
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[mem.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[mem.cpp](#)

5.13 Monitor Class Reference

```
#include <monitor.h>
```

Collaboration diagram for Monitor:



Public Member Functions

- `Monitor (MainController *)`
- `~Monitor ()`
- `void rx_write ()`
- `void rx_status_read ()`
- `void rx_status_write ()`

- void `rx_read()`
- void `rx_erase()`
- void `rx_nuke()`
- void `rx_start_motor()`
- void `rx_stop_motor()`
- void `rx_go_active()`
- void `rx_test_read()`
- void `rx_set_serial()`
- void `rx_set_maxi()`
- void `rx_set_date()`
- void `rx_scan(void)`
- void `rx_scan(uint8_t cs_num, uint32_t address)`
- void `rx_get_man_id()`
- void `rx_find_pulses()`
- void `rx_upload()`
- void `rx_debug()`
- void `rx_exit()`
- void `rx_help()`
- void `rx_pass()`
- void `serial_consume()`

Public Attributes

- `uint8_t buffer[RXBUFFERSIZE]`
- `uint32_t address`
- `uint32_t value`
- `uint8_t chip_select`
- `MainController * apt`

5.13.1 Detailed Description

Definition at line 7 of file monitor.h.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 Monitor()

```
Monitor::Monitor (
    MainController * apt )
```

Definition at line 6 of file monitor.cpp.

```
7 {
8     this->apt = apt;
9 }
```

5.13.2.2 ~Monitor()

```
Monitor::~Monitor ( )
```

Definition at line 10 of file monitor.cpp.

```
11 {
12
13 }
```

5.13.3 Member Function Documentation

5.13.3.1 rx_debug()

```
void Monitor::rx_debug ( )
```

Definition at line 346 of file monitor.cpp.

```
347 {
348     this->apt->state->next = UartDebug;
349     this->apt->state->current = UartDebug;
350 }
```

Here is the caller graph for this function:



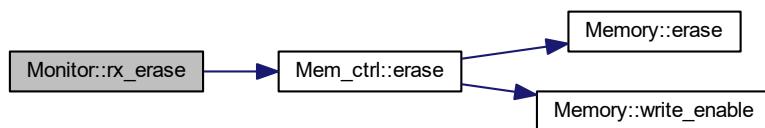
5.13.3.2 rx_erase()

```
void Monitor::rx_erase ( )
```

Definition at line 185 of file monitor.cpp.

```
186 {
187     if (this->tkn_i > 2)
188     {
189         uint8_t cs_num = *(this->tokens[1]) - '0';      // try to parse into cs number
190         uint32_t value = strtoul(this->tokens[2], NULL, 16);
191         this->apt->mem->erase(cs_num, value);
192     }
193 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.3 rx_exit()

```
void Monitor::rx_exit ( )
```

Definition at line 352 of file monitor.cpp.

```
353 {
354     NVIC_SystemReset ();
355 }
```

Here is the caller graph for this function:



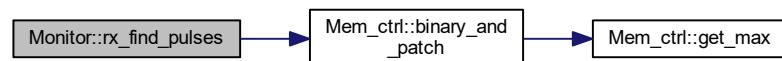
5.13.3.4 rx_find_pulses()

```
void Monitor::rx_find_pulses ( )
```

Definition at line 328 of file monitor.cpp.

```
329 {
330     if (this->tkn_i > 1)
331     {
332         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
333         uint32_t time_delay = HAL_GetTick ();
334         this->apt->mem->binary_and_patch ();
335         uint32_t pulse_count = *this->apt->pulses;
336         char pulse_str[40];
337         sprintf(pulse_str, "\r\nFound: %lu pulses in %lu time", pulse_count, HAL_GetTick () -
338             time_delay);
338         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)pulse_str, strlen(pulse_str), 3);
339     }
340 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.5 rx_get_man_id()

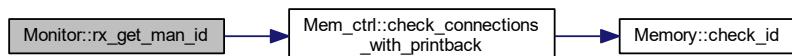
```
void Monitor::rx_get_man_id ( )
```

Definition at line 320 of file monitor.cpp.

```

321 {
322     if (this->tkn_i > 1)
323     {
324         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
325         this->apt->mem->check_connections_with_printback(cs_num);
326     }
327 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.6 rx_go_active()

```
void Monitor::rx_go_active ( )
```

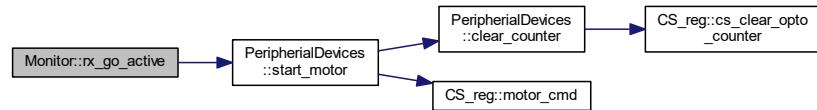
Definition at line 237 of file monitor.cpp.

```

238 {
239     this->apt->device->start_motor();
240     this->apt->pulses_on_screen = 0;
241     this->apt->state->next = Active;
```

```
242     this->apt->state->current = Active;
243 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.7 rx_help()

```
void Monitor::rx_help ( )
```

Definition at line 357 of file monitor.cpp.

```
358 {
359     const char help_str[] = \
360     "\r\n=====\
361     "\r\n= Usage:\
362     "\r\n= help# - this message\
363     "\r\n= exit# - return to normal operation\
364     "\r\n= debug# - halt operation and read uart only\
365     "\r\n= rd# - read arbitrary memory at chipselect\
366     "\r\n=         ex. rd,3,0x000ffff0#\
367     "\r\n= wrt# - write arbitrary memory at chipselect\
368     "\r\n=         ex. wrt,3,0x000ffff0,10#\
369     "\r\n= erase# - erase 4K block at loaction ex. erase,3,0x1000#\
370     "\r\n= nuke# - erase whole chip wholly\
371     "\r\n= upload# - Xmodem transfer of data\
372     "\r\n= scan# - read 256 bytes\
373     "\r\n= status1/status2# - read status register\
374     "\r\n= statusw# - write status register\
375     "\r\n= M=1#/M=0 - run/halt motor\
376     "\r\n= pass# - pass command to screen\
377     "\r\n= snum,date,maxi, testread           - header setters\
378     "\r\n=====\
379     "\r\n\r\n";\
380     HAL_UART_Transmit (SERIAL_PC, (uint8_t*)help_str, strlen(help_str), 10);\
381 }
382 }
```

Here is the caller graph for this function:



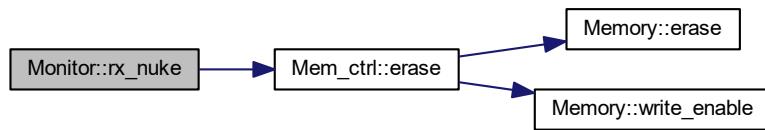
5.13.3.8 rx_nuke()

```
void Monitor::rx_nuke ( )
```

Definition at line 176 of file monitor.cpp.

```
177 {
178     if (this->tkn_i > 0)
179     {
180         uint8_t cs_num = *(this->tokens[1]) - '0'; // try to parse into cs number
181         this->apt->mem->erase(cs_num);
182     }
183 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.9 rx_pass()

```
void Monitor::rx_pass ( )
```

Definition at line 383 of file monitor.cpp.

```
384 {
385     if (this->tkn_i > 0)
386     {
387         this->apt->screen->pass_to_ard((uint8_t*)this->tokens[1]);
388     }
389 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.10 rx_read()

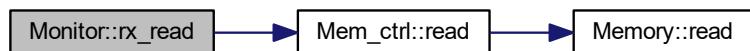
```
void Monitor::rx_read ( )
```

Definition at line 212 of file monitor.cpp.

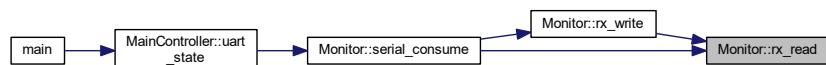
```

213 {
214     if (this->tkn_i > 1)
215     {
216         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
217         uint32_t address = strtoul(this->tokens[2], NULL, 16);
218         uint32_t value = strtoul(this->tokens[3], NULL, 16);
219         uint32_t read_back = this->apt->mem->read(cs_num, address);
220
221         char read_str[40];
222         sprintf(read_str, "\r\nRead: %08lX at addr %08lX\r\n", read_back, address);
223         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)read_str, strlen(read_str), 10);
224     }
225 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.11 rx_scan() [1/2]

```
void Monitor::rx_scan (
    uint8_t cs_num,
    uint32_t address )
```

Definition at line 315 of file monitor.cpp.

```
316 {
317     this->apt->mem->read256(cs_num, address);
318 }
```

Here is the call graph for this function:



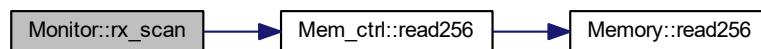
5.13.3.12 rx_scan() [2/2]

```
void Monitor::rx_scan (
    void )
```

Definition at line 305 of file monitor.cpp.

```
306 {
307     if (this->tkn_i > 2)
308     {
309         uint8_t cs_num = *(this->tokens[1]) - '0';
310         uint32_t token_numeric = strtoul(this->tokens[2], NULL, 16);
311         this->apt->mem->read256(cs_num, token_numeric);
312     }
313 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



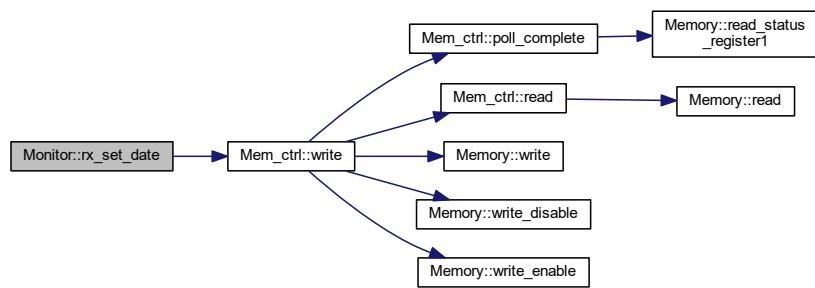
5.13.3.13 rx_set_date()

```
void Monitor::rx_set_date ( )
```

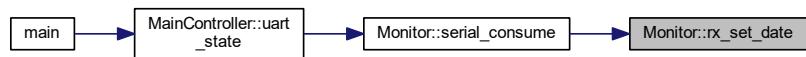
Definition at line 294 of file monitor.cpp.

```
295 {
296     if (this->tkn_i > 2)
297     {
298         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
299         uint32_t token_numeric = strtoul(this->tokens[2], NULL, 16);
300
301         this->apt->mem->write(cs_num, 0x00FFFF2, 0x44415445);
302         this->apt->mem->write(cs_num, 0x00FFEE, token_numeric);
303     }
304 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



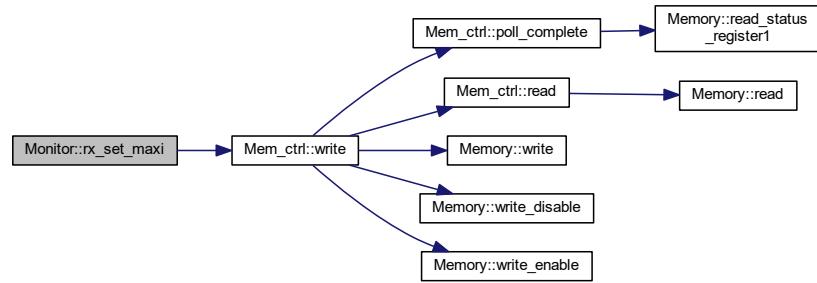
5.13.3.14 rx_set_maxi()

```
void Monitor::rx_set_maxi ( )
```

Definition at line 282 of file monitor.cpp.

```
283 {
284     if (this->tkn_i > 2)
285     {
286         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
287         uint32_t token_numeric = strtoul(this->tokens[2], NULL, 16);
288
289         this->apt->mem->write(cs_num, 0x00FFFEA, 0x4D415849);
290         this->apt->mem->write(cs_num, 0x00FFFE6, token_numeric);
291     }
292 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.15 rx_set_serial()

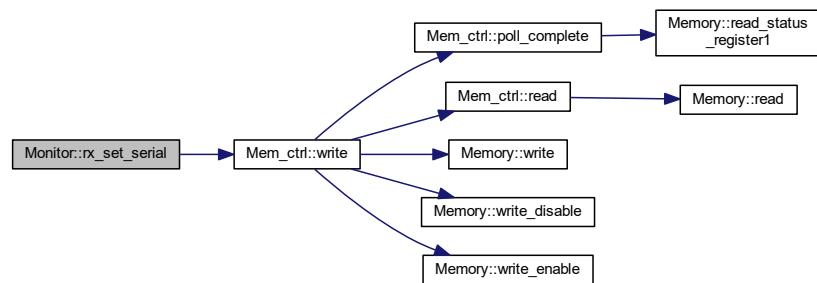
```
void Monitor::rx_set_serial ( )
```

Definition at line 271 of file monitor.cpp.

```

272 {
273     if (this->tkn_i > 2)
274     {
275         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
276         uint32_t token_numeric = strtoul(this->tokens[2], NULL, 16);
277
278         this->apt->mem->write(cs_num, 0x00FFFFA, 0x534E554D);
279         this->apt->mem->write(cs_num, 0x00FFFF6, token_numeric);
280     }
281 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



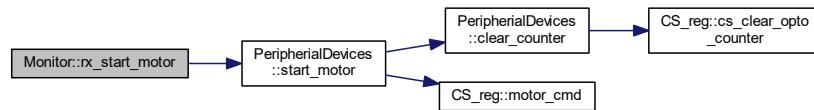
5.13.3.16 rx_start_motor()

```
void Monitor::rx_start_motor ( )
```

Definition at line 227 of file monitor.cpp.

```
228 {
229     this->apt->device->start_motor();
230 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



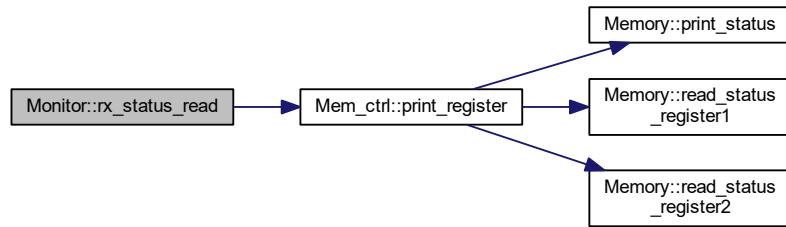
5.13.3.17 rx_status_read()

```
void Monitor::rx_status_read ( )
```

Definition at line 156 of file monitor.cpp.

```
157 {
158     if (this->tkn_i > 0)
159     {
160         uint8_t cs_num = *(this->tokens[1]) - '0'; // try to parse into cs number
161         this->apt->mem->print_register(cs_num);
162     }
163 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.18 rx_status_write()

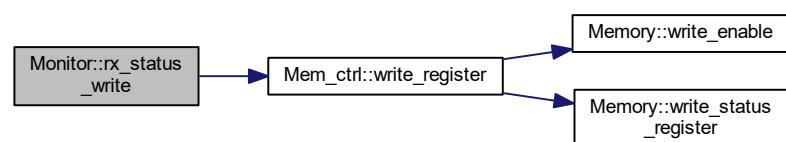
```
void Monitor::rx_status_write( )
```

Definition at line 166 of file monitor.cpp.

```

167 {
168     if (this->tkn_i > 1)
169     {
170         uint8_t cs_num = *(this->tokens[1]) - '0'; // try to parse into cs number
171         uint16_t value = strtoul(this->tokens[2], NULL, 16);
172         this->apt->mem->write_register(cs_num, value);
173     }
174 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



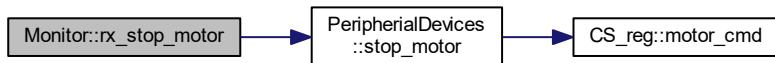
5.13.3.19 rx_stop_motor()

```
void Monitor::rx_stop_motor( )
```

Definition at line 232 of file monitor.cpp.

```
233 {
234     this->apt->device->stop_motor();
235 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.20 rx_test_read()

```
void Monitor::rx_test_read( )
```

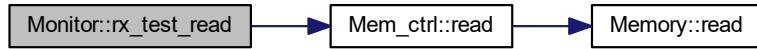
Definition at line 245 of file monitor.cpp.

```
246 {
247     if (this->tkn_i > 2)
248     {
249         uint8_t cs_num = *(this->tokens[1]) - '0';
250
251         int t_snum = this->apt->mem->read(cs_num, 0x00FFFFA);
252         int t_snum_v = this->apt->mem->read(cs_num, 0x00FFFF6);
```

```

253     int t_date = this->apt->mem->read(cs_num, 0x00FFFF2);
254     int t_date_v = this->apt->mem->read(cs_num, 0x00FFEE);
255     int t_max = this->apt->mem->read(cs_num, 0x00FFEA);
256     int t_max_v = this->apt->mem->read(cs_num, 0x00FFE6);
257
258     char ap_status[100];
259
260     char snum[5];
261     this->translate_int_to_arr(t_snum, (uint8_t*)snum);
262     char date_t[5];
263     this->translate_int_to_arr(t_date, (uint8_t*)date_t);
264     char maxi[5];
265     this->translate_int_to_arr(t_max, (uint8_t*)maxi);
266
267     sprintf(ap_status, "\r\n%s %08x\r\n%s %d\r\n%s %d\r\n", snum, t_snum_v, date_t, t_date_v, maxi,
268             t_max_v);
268     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)ap_status, strlen(ap_status), 100);
269 }
270 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.21 rx_upload()

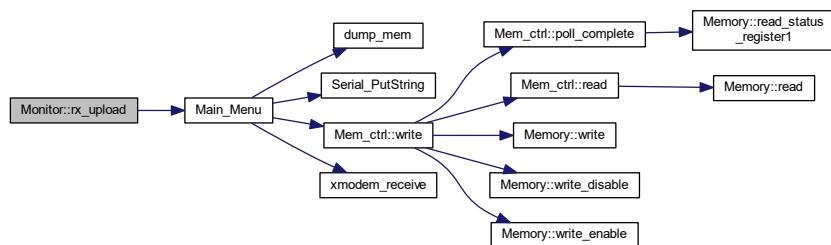
```
void Monitor::rx_upload( )
```

Definition at line 342 of file monitor.cpp.

```

343 {
344     Main_Menu();
345 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.22 rx_write()

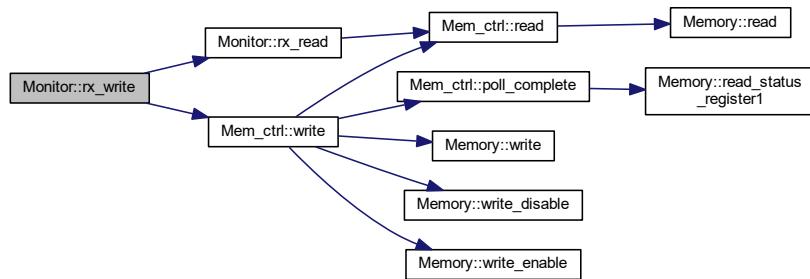
```
void Monitor::rx_write ( )
```

Definition at line 195 of file monitor.cpp.

```

196 {
197     if (this->tkn_i > 3)
198     {
199         uint8_t cs_num = *(this->tokens[1]) - '0';           // try to parse into cs number
200         uint32_t address = strtoul(this->tokens[2], NULL, 16);
201         uint32_t value = strtoul(this->tokens[3], NULL, 16);
202         this->apt->mem->write(cs_num, address, value);
203
204         char write_str[40];
205         sprintf(write_str, "\r\nWrite: %08lX at addr %08lX\r\n", value, address);
206         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)write_str, strlen(write_str), 10);
207         HAL_Delay(1);
208         this->rx_read();
209     }
210 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.3.23 serial_consume()

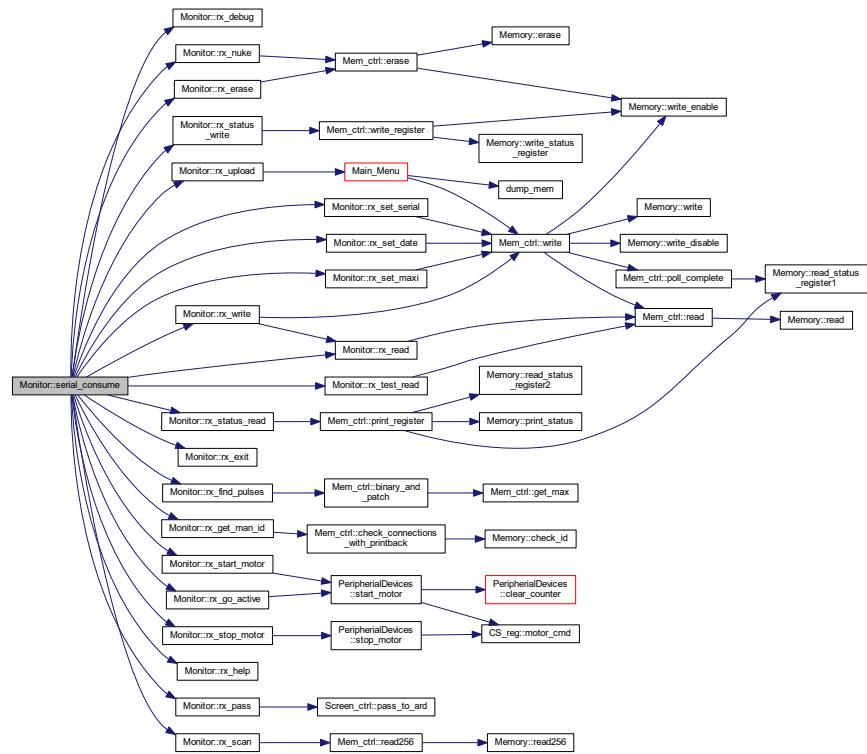
```
void Monitor::serial_consume (
    void )
```

Definition at line 14 of file monitor.cpp.

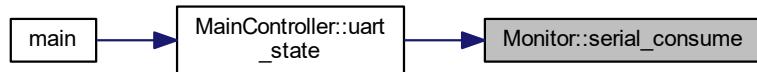
```
15 {
16     HAL_StatusTypeDef status;
17     status = HAL_UART_Receive(SERIAL_PC, (uint8_t*)&this->buffer[this->index], 1, 10);
18     if (status == HAL_OK)
19     {
20         if (rx_char == '%' || rx_char == '#')
21         {
22             buffer[index] = 0;
23             index = 0;
24             const char brk[] = "\r\n";
25             HAL_UART_Transmit(SERIAL_PC, (uint8_t*)brk, strlen(brk), 10);
26             HAL_UART_Transmit(SERIAL_PC, this->buffer, strlen((char*)this->buffer), 100);
27             HAL_UART_Transmit(SERIAL_PC, (uint8_t*)brk, strlen(brk), 10);
28             char* rest = (char*)this->buffer;
29             // Returns first token
30             this->tkn_i = 0;
31             this->tokens[0] = strtok_r(rest, ",", &rest);
32
33             // Keep printing tokens while one of the
34             // delimiters present in str[].
35             while(this->tokens[tkn_i] != NULL)
36             {
37                 this->tkn_i++;
38                 this->tokens[tkn_i] = strtok_r(rest, ",", &rest);
39             }
40             if (this->tokens[0] != NULL)
41             {
42                 // PARSE THE STRING
43                 if(strncmp(this->tokens[0], "wrt", 3) == 0)
44                 {
45                     this->rx_write();
46                 }
47                 else if(strncmp(this->tokens[0], "status1", 7) == 0)
48                 {
49                     this->rx_status_read();
50                 }
51                 else if(strncmp(this->tokens[0], "statusw", 7) == 0)
52                 {
53                     this->rx_status_write();
54                 }
55                 else if(strncmp(this->tokens[0], "rd", 2) == 0)
56                 {
57                     this->rx_read();
58                 }
59                 else if(strncmp(this->tokens[0], "erase", 5) == 0)
60                 {
61                     this->rx_erase();
62                 }
63                 else if(strncmp(this->tokens[0], "nuke", 4) == 0)
64                 {
65                     this->rx_nuke();
66                 }
67                 else if(strncmp(this->tokens[0], "M=1", 3) == 0)
68                 {
69                     this->rx_start_motor();
70                 }
71                 else if(strncmp(this->tokens[0], "M=0", 3) == 0)
72                 {
73                     this->rx_stop_motor();
74                 }
75                 else if(strncmp(this->tokens[0], "RUN", 3) == 0)
76                 {
77                     this->rx_go_active();
78                 }
79                 else if (strncmp(this->tokens[0], "testread", 8) == 0)
80                 {
81                     this->rx_test_read();
82                 }
83                 else if (strncmp(this->tokens[0], "snum", 4) == 0)
84                 {
85                     this->rx_set_serial();
86                 }
87                 else if (strncmp(this->tokens[0], "maxi", 4) == 0)
88                 {
89                     this->rx_set_maxi();
90                 }
91                 else if (strncmp(this->tokens[0], "date", 4) == 0)
92                 {
```

```
93         this->rx_set_date();
94     }
95     else if (strncmp(this->tokens[0], "scan", 4) == 0)
96     {
97         this->rx_scan();
98     }
99     else if (strncmp(this->tokens[0], "dump", 4) == 0)
100    {
101        for (uint32_t i = 0; i < 0x100000 / 0x100; i++)
102        {
103            this->rx_scan(3, i * 0x100);
104        }
105    }
106    else if (strncmp(this->tokens[0], "flashread", 9) == 0)
107    {
108        // flashread
109        // 012345678
110        for(uint32_t i = 0 ; i < 0x100000 / 0x100 ; i++)
111        {
112            this->rx_scan(1, i * 0x100);
113        }
114    }
115    else if (strncmp(this->tokens[0], "getid", 5) == 0)
116    {
117        this->rx_get_man_id();
118    }
119    else if (strncmp(this->tokens[0], "find", 4) == 0)
120    {
121        this->rx_find_pulses();
122    }
123    else if (strncmp(this->tokens[0], "upload", 5) == 0)
124    {
125        this->rx_upload();
126    }
127    else if (strncmp(this->tokens[0], "debug", 5) == 0)
128    {
129        this->rx_debug();
130    }
131    else if (strncmp(this->tokens[0], "exit", 4) == 0)
132    {
133        this->rx_exit();
134    }
135    else if (strncmp(this->tokens[0], "help", 4) == 0)
136    {
137        this->rx_help();
138    }
139    else if (strncmp(this->tokens[0], "pass", 4) == 0)
140    {
141        this->rx_pass();
142    }
143}
144}
145}
146}
147else
148{
149    this->buffer[this->index] = rx_char ;
150    this->index++;
151    this->index = this->index % RXBUFSIZE;
152}
153}
154}
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.13.4 Member Data Documentation

5.13.4.1 address

```
uint32_t Monitor::address
```

Definition at line 11 of file monitor.h.

5.13.4.2 apt

```
MainController* Monitor::apt
```

Definition at line 14 of file monitor.h.

5.13.4.3 buffer

```
uint8_t Monitor::buffer[RXBUFFERSIZE]
```

Definition at line 10 of file monitor.h.

5.13.4.4 chip_select

```
uint8_t Monitor::chip_select
```

Definition at line 13 of file monitor.h.

5.13.4.5 value

```
uint32_t Monitor::value
```

Definition at line 12 of file monitor.h.

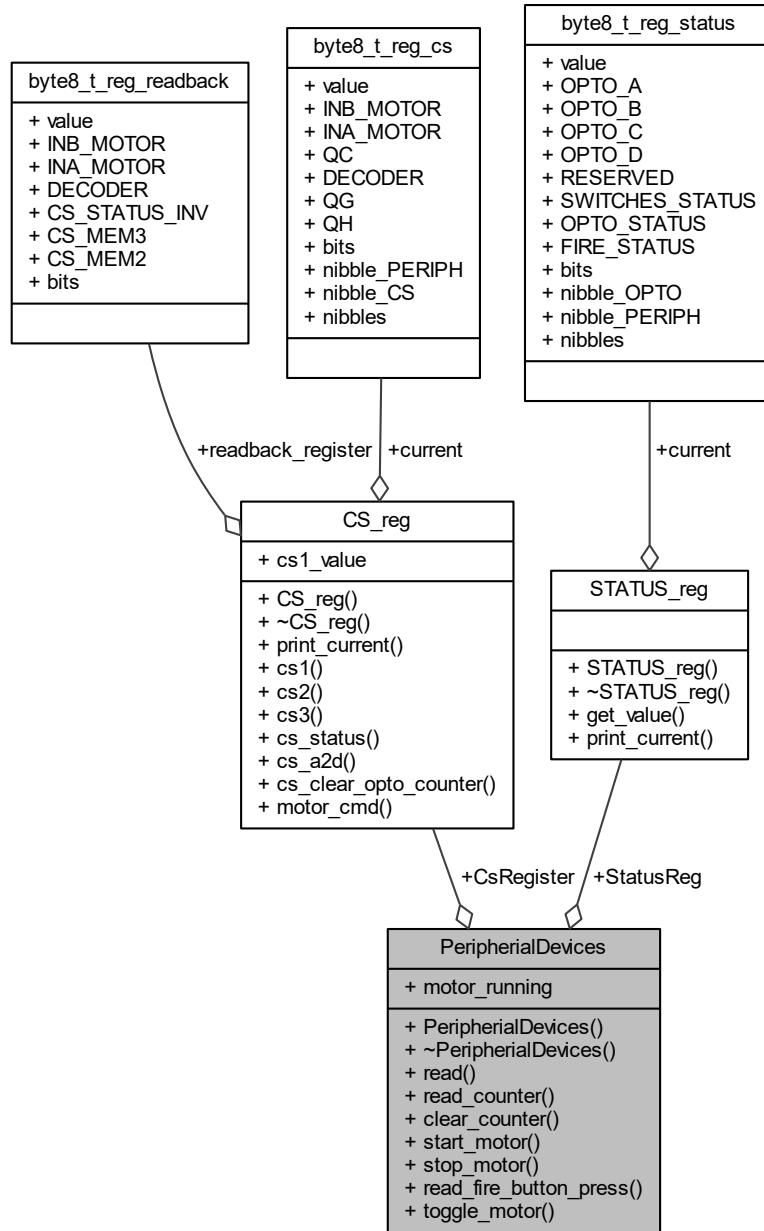
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.h
- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.cpp

5.14 PeripheralDevices Class Reference

```
#include <peripheral_controller.h>
```

Collaboration diagram for PeripheralDevices:



Public Member Functions

- `PeripheralDevices (CS_reg *cs, STATUS_reg *status)`
- `~PeripheralDevices ()`

- void `read` (void)
- uint32_t `read_counter` (void)
- void `clear_counter` (void)
- void `start_motor` (void)
- void `stop_motor` (void)
- bool `read_fire_button_press` (void)
- bool `toggle_motor` (void)

Public Attributes

- `CS_reg * CsRegister`
- `STATUS_reg * StatusReg`
- bool `motor_running`

5.14.1 Detailed Description

Definition at line 4 of file peripheral_controller.h.

5.14.2 Constructor & Destructor Documentation

5.14.2.1 PeripheralDevices()

```
PeripheralDevices::PeripheralDevices (
    CS_reg * cs,
    STATUS_reg * status )
```

Definition at line 5 of file peripheral_controller.cpp.

```
6 {
7     this->CsRegister = cs;
8     this->StatusReg = status;
9 }
```

5.14.2.2 ~PeripheralDevices()

```
PeripheralDevices::~PeripheralDevices ( )
```

Definition at line 11 of file peripheral_controller.cpp.

```
12 {
13
14 }
```

5.14.3 Member Function Documentation

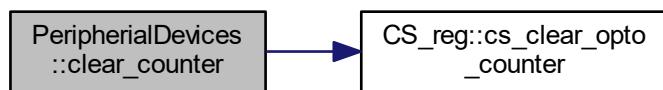
5.14.3.1 clear_counter()

```
void PeripheralDevices::clear_counter (
    void )
```

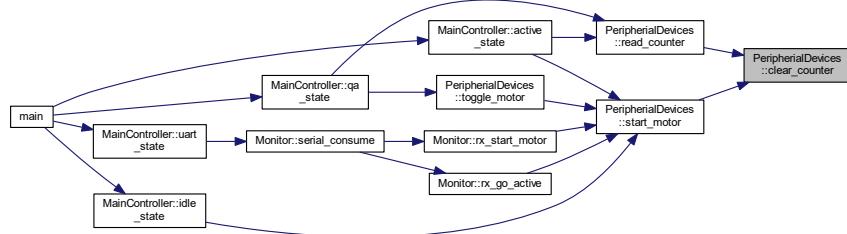
Definition at line 23 of file peripheral_controller.cpp.

```
24 {
25     this->CsRegister->cs_clear_opto_counter(LOW);
26     HAL_Delay(1);
27 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



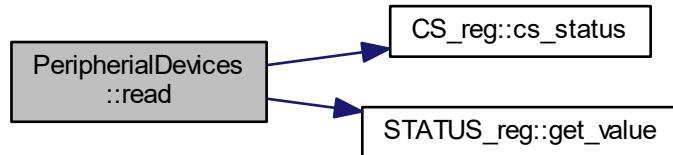
5.14.3.2 read()

```
void PeripheralDevices::read (
    void )
```

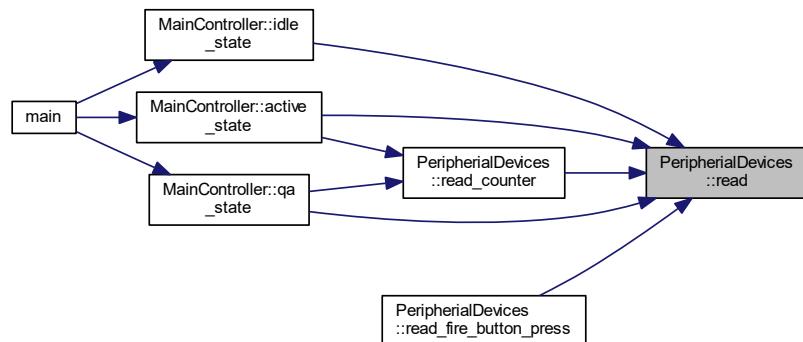
Definition at line 16 of file peripheral_controller.cpp.

```
17 {
18     this->CsRegister->cs_status(LOW);
19     this->StatusReg->get_value();
20     this->CsRegister->cs_status(HIGH);
21 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



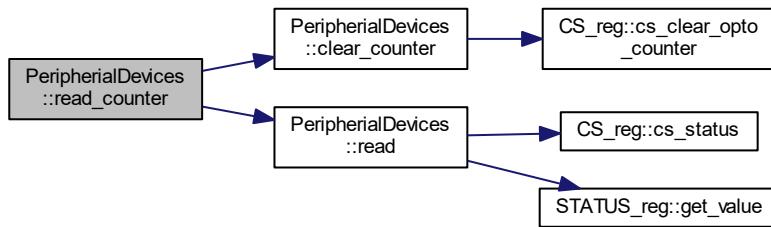
5.14.3.3 `read_counter()`

```
uint32_t PeripheralDevices::read_counter (
    void )
```

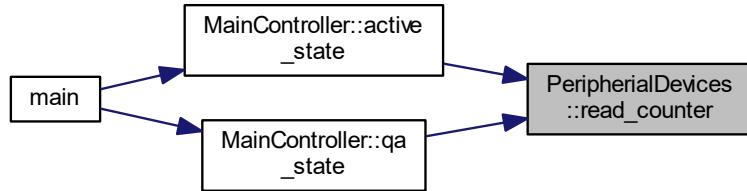
Definition at line 28 of file `peripheral_controller.cpp`.

```
29 {
30     this->read();
31     HAL_Delay(1);
32     this->clear_counter();
33     return this->StatusReg->current.nibbles.nibble_OPTO;
34 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



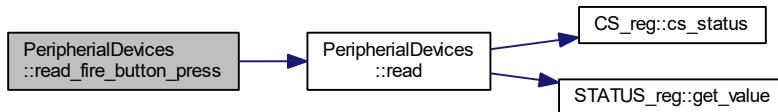
5.14.3.4 `read_fire_button_press()`

```
bool PeripheralDevices::read_fire_button_press (
    void )
```

Definition at line 61 of file peripheral_controller.cpp.

```
62 {
63     this->read();
64     return this->StatusReg->current.bits.FIRE_STATUS;
65 }
```

Here is the call graph for this function:



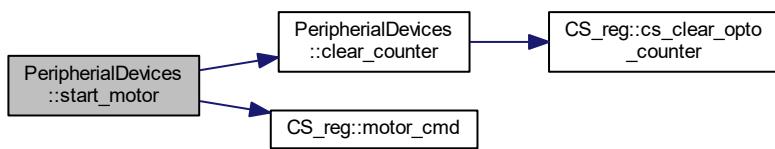
5.14.3.5 start_motor()

```
void PeripheralDevices::start_motor (
    void )
```

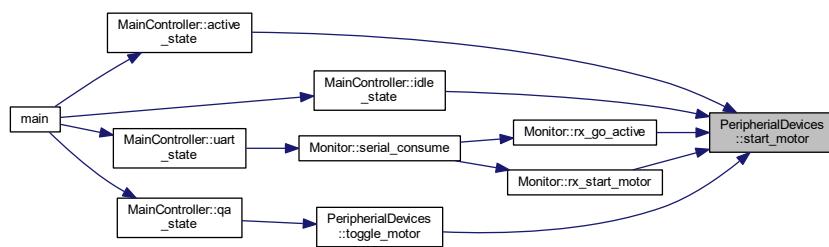
Definition at line 36 of file peripheral_controller.cpp.

```
37 {
38     this->clear_counter();
39     this->CsRegister->motor_cmd(true);
40     this->motor_running = true;
41 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



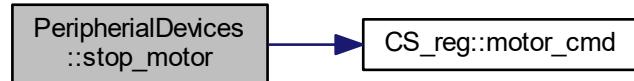
5.14.3.6 stop_motor()

```
void PeripheralDevices::stop_motor (
    void )
```

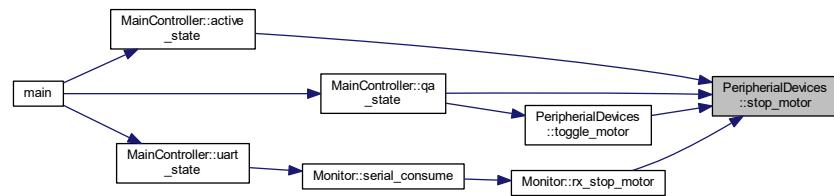
Definition at line 43 of file peripheral_controller.cpp.

```
44 {
45     this->CsRegister->motor_cmd(false);
46     this->motor_running = false;
47 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



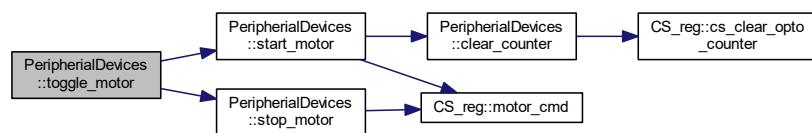
5.14.3.7 toggle_motor()

```
bool PeripheralDevices::toggle_motor (
    void )
```

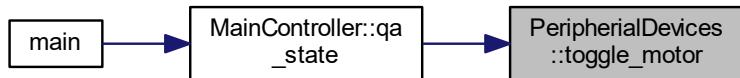
Definition at line 49 of file peripheral_controller.cpp.

```
50 {
51     if (this->motor_running)
52     {
53         this->stop_motor();
54     }
55     else
56     {
57         this->start_motor();
58     }
59     return this->motor_running;
60 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



5.14.4 Member Data Documentation

5.14.4.1 CsRegister

```
CS_REG* PeripheralDevices::CsRegister
```

Definition at line 7 of file peripheral_controller.h.

5.14.4.2 motor_running

```
bool PeripheralDevices::motor_running
```

Definition at line 9 of file peripheral_controller.h.

5.14.4.3 StatusReg

```
STATUS_REG* PeripheralDevices::StatusReg
```

Definition at line 8 of file peripheral_controller.h.

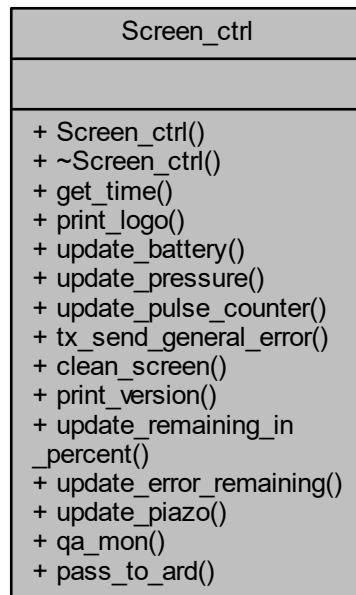
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[peripheral_controller.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[peripheral_controller.cpp](#)

5.15 Screen_ctrl Class Reference

```
#include <screen_controller.h>
```

Collaboration diagram for Screen_ctrl:



Public Member Functions

- `Screen_ctrl ()`
- `~Screen_ctrl ()`
- `void get_time (void)`
time output
- `void print_logo (void)`
print logo of armenta
- `void update_battery (int16_t bat_adc, uint32_t battery)`
update the battery
- `void update_pressure (uint32_t pressure)`
update the pressure
- `void update_pulse_counter (uint32_t pulses)`
update the pulses
- `void tx_send_general_error (uint32_t reason)`
send error
- `void clean_screen (uint32_t new_pulses_to_screen)`
reset screen state
- `void print_version (void)`
show version on screen

- void [update_remaining_in_percent](#) (uint32_t max_pulses, uint32_t pulses)
show percentage of AM usage on screen
- void [update_error_remaining](#) (uint32_t remaining)
show warning of remaining less than 1000
- void [update_piazo](#) (bool piezo)
update pulse symbol on screen
- void [qa_mon](#) (bool valid_mcu, bool valid_apt, bool valid_am)
update qa status to screen
- void [pass_to_ard](#) (uint8_t *message)

5.15.1 Detailed Description

Definition at line 4 of file screen_controller.h.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 Screen_ctrl()

```
Screen_ctrl::Screen_ctrl ( )
```

Definition at line 6 of file screen_controller.cpp.

```
7 {
8     this->last_remaining_message_sent = -1;
9     this->last_blocking_message = 0;
10    this->times_sent = 0;
11    memset(this->buffer, 0, sizeof(uint8_t)*RXBUFFERSIZE);
12    memset(this->buffer_with_crc, 0, sizeof(uint8_t)*RXBUFFERSIZE);
13 }
```

5.15.2.2 ~Screen_ctrl()

```
Screen_ctrl::~Screen_ctrl ( )
```

Definition at line 15 of file screen_controller.cpp.

```
16 {
17
18 }
```

5.15.3 Member Function Documentation

5.15.3.1 clean_screen()

```
void Screen_ctrl::clean_screen (
    uint32_t new_pulses_to_screen )
```

reset screen state

sometimes we need to be certain the screen is cleared

Definition at line 211 of file screen_controller.cpp.

```
212 {
213     sprintf((char*)this->buffer, "$Q%lu#", new_pulses_to_screen);
214     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
215     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
216 }
```

5.15.3.2 get_time()

```
void Screen_ctrl::get_time (
    void )
```

time output

Print time from cpu start to serial

Definition at line 63 of file screen_controller.cpp.

```
64 {
65     RTC_DateTypeDef sdatestructureget;
66     RTC_TimeTypeDef stimestructureget;
67     HAL_RTC_GetTime(&hrtc, &stimestructureget, RTC_FORMAT_BIN);
68     /* Get the RTC current Date */
69     HAL_RTC_GetDate(&hrtc, &sdatestructureget, RTC_FORMAT_BIN);
70     /* Display time Format : hh:mm:ss */
71
72     char aShowTime[100];
73     sprintf((char*)this->buffer,
74         "Time %02d:%02d:%02d \r\n",
75         stimestructureget.Hours,
76         stimestructureget.Minutes,
77         stimestructureget.Seconds);
78     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 100);
79 }
```

5.15.3.3 pass_to_ard()

```
void Screen_ctrl::pass_to_ard (
    uint8_t * message )
```

Definition at line 331 of file screen_controller.cpp.

```
332 {
333     sprintf((char*)this->buffer, "%s#", (char*)message);
334     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
335     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
336 }
```

Here is the caller graph for this function:



5.15.3.4 print_logo()

```
void Screen_ctrl::print_logo (
    void )
```

print logo of armenta

Its a cow

Definition at line 81 of file screen_controller.cpp.

```
82 {
83     const uint8_t logo[] = ""
84     " NNNN      NNNN  \r\n"
85     " NNNNiiiiiiiiiiiiiiiiiiNNNN  \r\n"
86     " NNNNNNNNNNNNNNNNNNNNNNNNNNN  \r\n"
87     "          iNNi  \r\n"
88     "      oooo  iNNi  oooo  \r\n"
89     "      pppp  iNNi  pppp  \r\n"
90     "          iNNi  \r\n"
91     "          iNNi  \r\n"
92     "          iNNi  \r\n"
93     "          iNNi  \r\n"
94     "          iNNi  \r\n"
95     "          iNNi  \r\n"
96     "          iNNi  \r\n"
97     "          iNNi  \r\n"
98     "          iNNi  \r\n";
99     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)logo, strlen((char*)logo), 100); // terminal
100    sprintf((char*)this->buffer, "ARMenta version: %s\r\n", VERSION);
101    HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 100); // terminal
102 }
```

5.15.3.5 print_version()

```
void Screen_ctrl::print_version (
    void )
```

show version on screen

sometimes we need to be certain the screen is cleared

Definition at line 218 of file screen_controller.cpp.

```
219 {
220     char version[] = VERSION;
221     sprintf((char*)this->buffer, "$v%s#", version);
222     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
223     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
224 }
```

5.15.3.6 qa_mon()

```
void Screen_ctrl::qa_mon (
    bool valid_mcu,
    bool valid_apt,
    bool valid_am )
```

update qa status to screen

Definition at line 303 of file screen_controller.cpp.

```
304 {
305
306     if (this->check_timer_expired())
307     {
308         sprintf((char*)this->buffer,
309                 "$Gcs1 %s \r\n"
310                 "cs2 %s\r\n"
311                 "cs3 %s\r\n",
312                 valid_mcu ? "true" : "false",
313                 valid_apt ? "true" : "false",
314                 valid_am ? "true" : "false");
315         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
316         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
317     }
318 }
```

Here is the caller graph for this function:



5.15.3.7 tx_send_general_error()

```
void Screen_ctrl::tx_send_general_error (
    uint32_t reason )
```

send error

when there is an error to show to the screen, we show a red screen and some text.

Definition at line 139 of file screen_controller.cpp.

```
140 {
141
142     if (this->check_timer_expired())
143     {
144         if (reason == 404) // MCU not reachable
145         {
146             sprintf((char*)this->buffer, "The Mcu is Unavailable, spi mem read failed\r\n");
147             HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
148             sprintf((char*)this->buffer, "$zMCU Unavailable#");
149             uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
150             HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
151         }
152         else if (reason == 405) // AP or body unreachable
153         {
154             sprintf((char*)this->buffer, "The APTx is Unavailable, spi mem read failed\r\n");
155         }
156     }
157 }
```

```

155         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
156         sprintf((char*)this->buffer, "$zAPTx Disconnected#");
157         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
158         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
159     }
160     else if (reason == 406) // AM
161     {
162         sprintf((char*)this->buffer, "The AM is Unavailable, spi mem read failed\r\n");
163         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
164         sprintf((char*)this->buffer, "$zAM Disconnected#");
165         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
166         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
167     }
168     else if (reason == 501)
169     {
170         sprintf((char*)this->buffer, "The AP is not initialized, please supply a valid s/n\r\n");
171         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
172         sprintf((char*)this->buffer, "$zAP sn not valid#");
173         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
174         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
175     }
176     else if (reason == 502)
177     {
178         sprintf((char*)this->buffer, "The AM cannot be written to\r\n");
179         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
180         sprintf((char*)this->buffer, "$zAM write error#");
181         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
182         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
183     }
184     else if (reason == 503)
185     {
186         sprintf((char*)this->buffer, "Device exceeded lifetime for Maintenance, will not run
187 anymore\r\n");
187         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
188         sprintf((char*)this->buffer, "$zError: Contact Maintenance E503#");
189         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
190         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
191     }
192     else if (reason == 504)
193     {
194         sprintf((char*)this->buffer, "Device near end of lifetime for Maintenance");
195         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
196         sprintf((char*)this->buffer, "$zWarning: Contact Maintenance E504#");
197         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
198         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
199     }
200     else if (reason == 200)
201     {
202         sprintf((char*)this->buffer, "Initial Run Complete\r\n");
203         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
204         sprintf((char*)this->buffer, "$zInitial Run Complete#");
205         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
206         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
207     }
208 }
209 }
```

Here is the caller graph for this function:



5.15.3.8 update_battery()

```

void Screen_ctrl::update_battery (
    int16_t bat_adc,
    uint32_t battery )
```

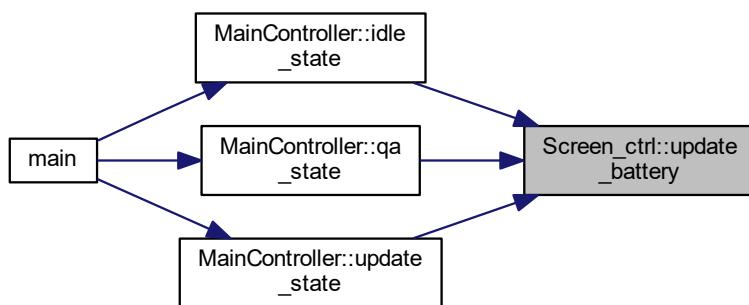
update the battery

update the battery on screen

Definition at line 104 of file screen_controller.cpp.

```
105 {
106     sprintf((char*)this->buffer, "BatteryADC:%d\r\nBatteryConversion:%lu\r\n", bat_adc, battery);
107     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 1000);
108
109     sprintf((char*)this->buffer, "$b%lu#", battery);
110     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
111     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
112 }
```

Here is the caller graph for this function:



5.15.3.9 update_error_remaining()

```
void Screen_ctrl::update_error_remaining (
    uint32_t remaining )
```

show warning of remaining less than 1000

we show warnings at:

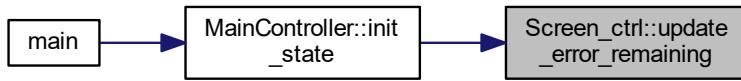
- 1000
- 800
- 600
- 400
- 200
- 0 - we stop if running and show if try to start

Definition at line 252 of file screen_controller.cpp.

```

253 {
254
255
256     if (this->last_remaining_message_sent != remaining)
257     {
258         this->times_sent = 0;
259         this->last_remaining_message_sent = remaining;
260     }
261
262
263
264     if (this->check_timer_expired())
265     {
266         // It is not another if condition, do not merge
267         if((remaining > 0) && (last_remaining_message_sent == remaining))
268         {
269             // By saving a static variable as memory of last sent $f[count]#
270             // We just send it [times_sent] times
271             if(this->times_sent < AM_WARNING_TIMES)
272             {
273                 sprintf((char*)this->buffer, "$f%lu#", remaining);
274                 uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
275                 HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
276                 this->times_sent++;
277             }
278         }
279         else if(remaining == 0)
280         {
281             sprintf((char*)this->buffer, "$f0#");
282             uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
283             HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
284         }
285     }
286     this->last_remaining_message_sent = remaining;
287 }
```

Here is the caller graph for this function:



5.15.3.10 update_piazo()

```
void Screen_ctrl::update_piazo (
    bool piezo )
```

update pulse symbol on screen

Currently we just show the green icon on motor running and red one when not. Should be determined by the piazo signal.

Definition at line 289 of file screen_controller.cpp.

```

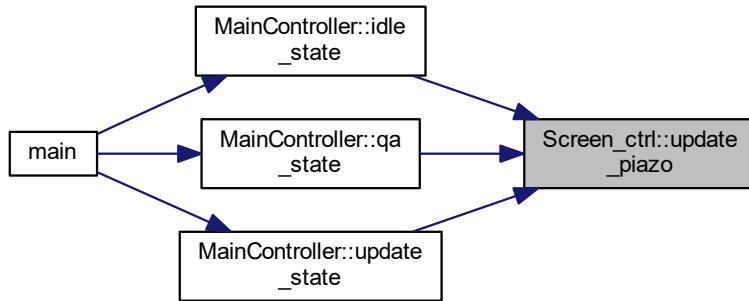
290 {
291     if (piezo)
292     {
293         sprintf((char*)this->buffer, "$aY#");
294     }
295     else
296     {
297         sprintf((char*)this->buffer, "$aN#");
```

```

298     }
299     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
300     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
301 }

```

Here is the caller graph for this function:



5.15.3.11 update_pressure()

```

void Screen_ctrl::update_pressure (
    uint32_t pressure )

```

update the pressure

update the pressure, if the pressure is red we cannot start the motor.

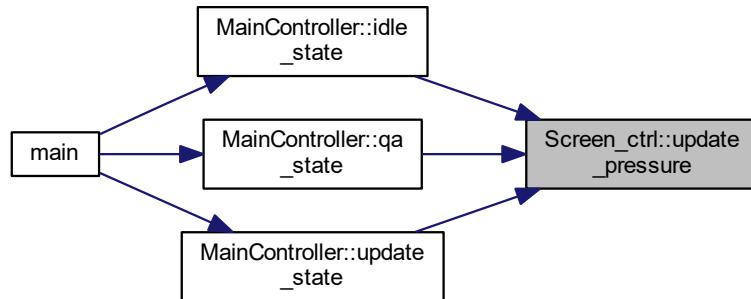
Definition at line 114 of file screen_controller.cpp.

```

115 {
116     if(pressure >= ADC_Presure_27_bar) // pressure too low
117     {
118         sprintf((char*)this->buffer, "Pr %lu [0.k]\r\n", pressure);
119         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 100);
120         sprintf((char*)this->buffer, "$pO.K#");
121     }
122     else
123     {
124         sprintf((char*)this->buffer, "Pr %lu [low]\r\n", pressure);
125         HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 100);
126         sprintf((char*)this->buffer, "$pLOW#");
127     }
128     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
129     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
130 }

```

Here is the caller graph for this function:



5.15.3.12 update_pulse_counter()

```
void Screen_ctrl::update_pulse_counter (
    uint32_t pulses )
```

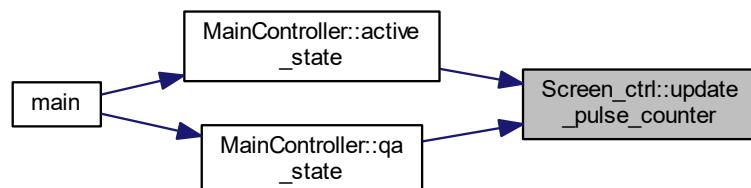
update the pulses

send the pulses on screen to the screen. This function has to be the quickest to respond.

Definition at line 132 of file screen_controller.cpp.

```
133 {
134     sprintf((char*)this->buffer, "$c%lu#", pulses);
135     uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
136     HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
137 }
```

Here is the caller graph for this function:



5.15.3.13 update_remaining_in_percent()

```
void Screen_ctrl::update_remaining_in_percent (
    uint32_t max_pulses,
    uint32_t pulses )
```

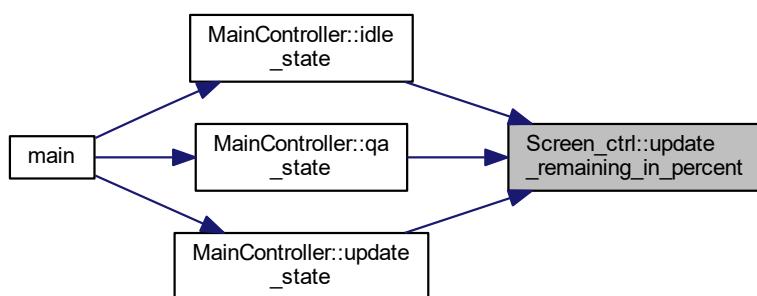
show percentage of AM usage on screen

we take the maximum - used / maximum *100

Definition at line 226 of file screen_controller.cpp.

```
227 {
228     uint32_t maxi = max_pulses;
229     uint32_t percent_remaining = (((maxi - pulses) * 100) / maxi);
230     uint32_t remainder_scaled = (percent_remaining * 11) / 10;
231     /*
232      * Management wanted some buffer to have Graphics show 100% until they use up some ammount of
233      * pulses
234      * Otherwise after 1 pulse it did ceil(99.9%) and showed 99%, which was not to their liking.
235      * So the function stretches 100% to 110% and then clips it at 100% thus giving 9% until you
236      * see 99%
237     */
238     if (remainder_scaled > 100)
239     {
240         remainder_scaled = 100;
241     }
242     if (max_pulses > 0)
243     {
244         sprintf((char*)this->buffer, "$t%lu#", remainder_scaled);
245         uint32_t size = this->prepare_packet(this->buffer, this->buffer_with_crc);
246         HAL_UART_Transmit(SERIAL_SCREEN, (uint8_t*)this->buffer_with_crc, size + 4, 10);
247     }
248     sprintf((char*)this->buffer, "pulses=%lu, max=%lu, util=%lu\r\n", pulses, max_pulses,
249     remainder_scaled);
250     HAL_UART_Transmit(SERIAL_PC, (uint8_t*)this->buffer, strlen((char*)this->buffer), 50);
250 }
```

Here is the caller graph for this function:



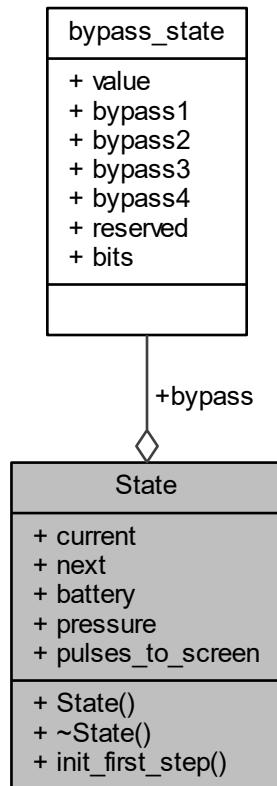
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[screen_controller.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[screen_controller.cpp](#)

5.16 State Class Reference

```
#include <state.h>
```

Collaboration diagram for State:



Public Member Functions

- [State \(\)](#)
- [~State \(\)](#)
- [void init_first_step \(void\)](#)

Public Attributes

- enum [statesMachine](#) `current`
- enum [statesMachine](#) `next`
- [bypass_state](#) `bypass`
- `uint32_t` `battery`
- `uint32_t` `pressure`
- `uint32_t` `pulses_to_screen`

5.16.1 Detailed Description

Definition at line 33 of file state.h.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 State()

```
State::State ( )
```

Definition at line 5 of file state.cpp.

```
6 {  
7     this->current = Init;  
8     this->next = Idle;  
9 }
```

5.16.2.2 ~State()

```
State::~State ( )
```

Definition at line 11 of file state.cpp.

```
12 {  
13  
14 }
```

5.16.3 Member Function Documentation

5.16.3.1 init_first_step()

```
void State::init_first_step (  
    void )
```

Definition at line 29 of file state.cpp.

```
30 {  
31     while (this->current == Init)  
32     {  
33         this->LOGIC_BYPASS_1 = (bool)HAL_GPIO_ReadPin(LOGIC_BYPASS_1_GPIO_Port, LOGIC_BYPASS_1_Pin);  
34         this->LOGIC_BYPASS_2 = (bool)HAL_GPIO_ReadPin(LOGIC_BYPASS_2_GPIO_Port, LOGIC_BYPASS_2_Pin);  
35         this->LOGIC_BYPASS_3 = (bool)HAL_GPIO_ReadPin(LOGIC_BYPASS_3_GPIO_Port, LOGIC_BYPASS_3_Pin);  
36         this->LOGIC_BYPASS_4 = (bool)HAL_GPIO_ReadPin(LOGIC_BYPASS_4_GPIO_Port, LOGIC_BYPASS_4_Pin);  
37         this->bypass.bits.bypass1 = this->LOGIC_BYPASS_1;  
38         this->bypass.bits.bypass2 = this->LOGIC_BYPASS_2;  
39         this->bypass.bits.bypass3 = this->LOGIC_BYPASS_3;  
40         this->bypass.bits.bypass4 = this->LOGIC_BYPASS_4;  
41         this->bypass.bits.reserved = 0;  
42     }  
43     if (this->bypass.value == 0)  
44     {  
45         read_interlock();  
46     }
```

```

48     else if (this->bypass.value == 1)
49     {
50         // don't connect the apt
51         this->current = Idle;
52     }
53     else if (this->bypass.value == 2)
54     {
55         // don't connect apt and start uart
56         this->current = UartDebug;
57     }
58 }
59 }
```

Here is the caller graph for this function:



5.16.4 Member Data Documentation

5.16.4.1 battery

```
uint32_t State::battery
```

Definition at line 39 of file state.h.

5.16.4.2 bypass

```
bypass_state State::bypass
```

Definition at line 38 of file state.h.

5.16.4.3 current

```
enum statesMachine State::current
```

Currently in this state.

Definition at line 36 of file state.h.

5.16.4.4 next

```
enum statesMachine State::next
```

Switch to this state next iteration.

Definition at line 37 of file state.h.

5.16.4.5 pressure

```
uint32_t State::pressure
```

Definition at line 40 of file state.h.

5.16.4.6 pulses_to_screen

```
uint32_t State::pulses_to_screen
```

Definition at line 41 of file state.h.

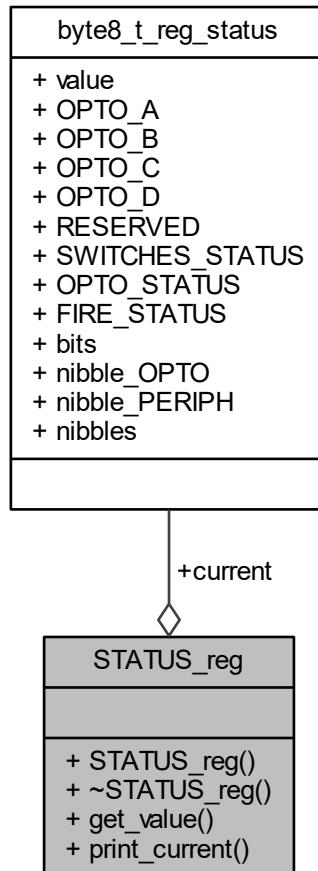
The documentation for this class was generated from the following files:

- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[state.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[state.cpp](#)

5.17 STATUS_reg Class Reference

```
#include <status_reg.h>
```

Collaboration diagram for STATUS_reg:



Public Member Functions

- [STATUS_reg \(\)](#)
- [~STATUS_reg \(\)](#)
- [uint8_t get_value \(void\)](#)
- [void print_current \(void\)](#)

Public Attributes

- [byte8_t_reg_status current](#)

5.17.1 Detailed Description

Definition at line 31 of file status_reg.h.

5.17.2 Constructor & Destructor Documentation

5.17.2.1 STATUS_reg()

```
STATUS_reg::STATUS_reg ( )
```

Definition at line 9 of file status_reg.cpp.

```
10 {  
11  
12 }
```

5.17.2.2 ~STATUS_reg()

```
STATUS_reg::~STATUS_reg ( )
```

Definition at line 14 of file status_reg.cpp.

```
15 {  
16  
17 }
```

5.17.3 Member Function Documentation

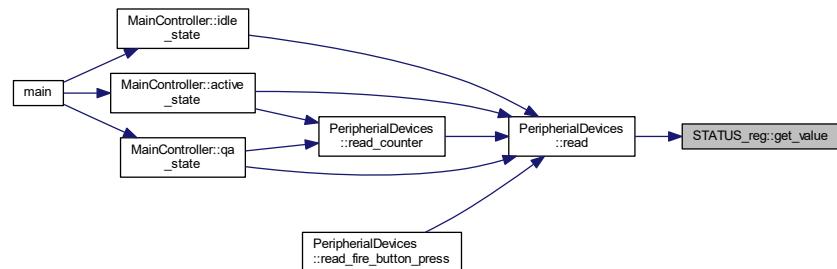
5.17.3.1 get_value()

```
uint8_t STATUS_reg::get_value ( void )
```

Definition at line 19 of file status_reg.cpp.

```
20 {  
21     HAL_SPI_Receive(&hspl1, &this->current.value, 1, 10);  
22     return this->current.value;  
23 }
```

Here is the caller graph for this function:



5.17.3.2 print_current()

```
void STATUS_reg::print_current (
    void )
```

Definition at line 25 of file status_reg.cpp.

```
26 {
27     uint8_t status[64];
28     sprintf((char*)status,
29             "[%d] OPTO_Count\r\n\
30             [%d] RESERVED\r\n\
31             [%d] SWITCHES\r\n\
32             [%d] OPTO\r\n\
33             [%d] FIRE\r\n",
34             this->current.nibbles.nibble_OPTO,
35             this->current.bits.RESERVED,
36             this->current.bits.SWITCHES_STATUS,
37             this->current.bits.OPTO_STATUS,
38             this->current.bits.FIRE_STATUS);
39     // TODO: consider decorator
40     HAL_UART_Transmit(SERIAL_PC, status, strlen((char*)status), 100);
41 }
```

5.17.4 Member Data Documentation

5.17.4.1 current

```
byte8_t_reg_status STATUS_reg::current
```

Definition at line 35 of file status_reg.h.

The documentation for this class was generated from the following files:

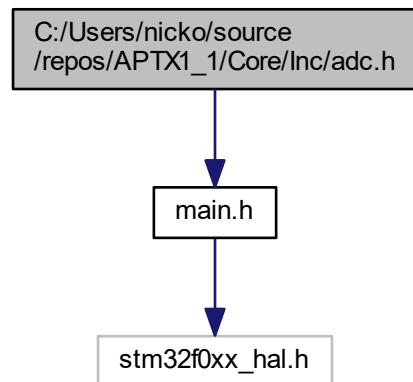
- C:/Users/nicko/source/repos/APTX1_1/Core/Inc/[status_reg.h](#)
- C:/Users/nicko/source/repos/APTX1_1/Core/Src/[status_reg.cpp](#)

Chapter 6

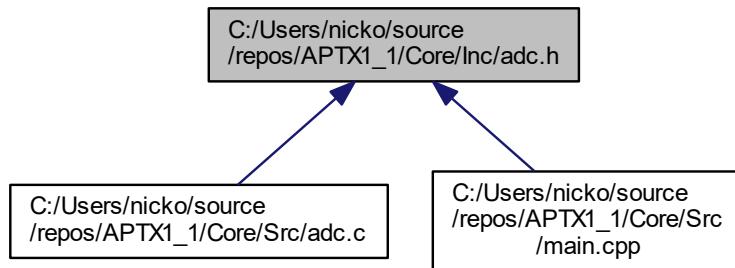
File Documentation

6.1 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc.h File Reference

```
#include "main.h"  
Include dependency graph for adc.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_ADC_Init](#) (void)

Variables

- ADC_HandleTypeDef [hadc](#)

6.1.1 Function Documentation

6.1.1.1 MX_ADC_Init()

```
void MX_ADC_Init (
    void )
```

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure for the selected ADC regular channel to be converted.

Configure for the selected ADC regular channel to be converted.

Configure for the selected ADC regular channel to be converted.

Configure for the selected ADC regular channel to be converted.

Definition at line 31 of file adc.c.

```
32 {
33     ADC_ChannelConfTypeDef sConfig = {0};
34
35     /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of
36     ** conversion)
37     */
38     hadc.Instance = ADC1;
39     hadc.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
40     hadc.Init.Resolution = ADC_RESOLUTION_10B;
```

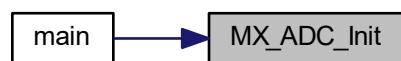
```

40     hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
41     hadc.Init.ScanConvMode = ADC_SCAN_DIRECTION_FORWARD;
42     hadc.Init.EOCSelection = ADC_EOC_SEQ_CONV;
43     hadc.Init.LowPowerAutoWait = DISABLE;
44     hadc.Init.LowPowerAutoPowerOff = DISABLE;
45     hadc.Init.ContinuousConvMode = ENABLE;
46     hadc.Init.DiscontinuousConvMode = DISABLE;
47     hadc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
48     hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
49     hadc.Init.DMAContinuousRequests = ENABLE;
50     hadc.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
51     if (HAL_ADC_Init(&hadc) != HAL_OK)
52     {
53         Error_Handler();
54     }
55     /** Configure for the selected ADC regular channel to be converted.
56     */
57     sConfig.Channel = ADC_CHANNEL_10;
58     sConfig.Rank = ADC_RANK_CHANNEL_NUMBER;
59     sConfig.SamplingTime = ADC_SAMPLETIME_239CYCLES_5;
60     if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
61     {
62         Error_Handler();
63     }
64     /** Configure for the selected ADC regular channel to be converted.
65     */
66     sConfig.Channel = ADC_CHANNEL_TEMPSENSOR;
67     if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
68     {
69         Error_Handler();
70     }
71     /** Configure for the selected ADC regular channel to be converted.
72     */
73     sConfig.Channel = ADC_CHANNEL_VREFINT;
74     if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
75     {
76         Error_Handler();
77     }
78     /** Configure for the selected ADC regular channel to be converted.
79     */
80     sConfig.Channel = ADC_CHANNEL_VBAT;
81     if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
82     {
83         Error_Handler();
84     }
85 }
86 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.1.2 Variable Documentation

6.1.2.1 hadc

```
ADC_HandleTypeDef hadc
```

File Name : [ADC.h](#) Description : This file provides code for the configuration of the ADC instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [ADC.c](#) Description : This file provides code for the configuration of the ADC instances.

Attention

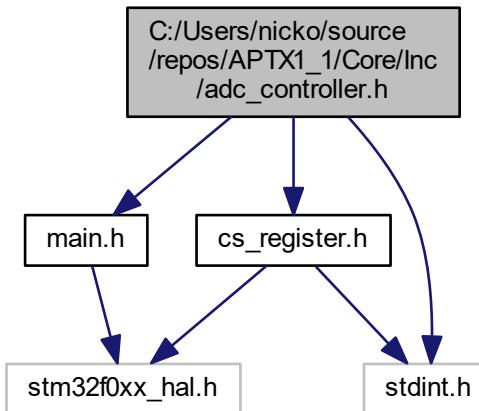
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

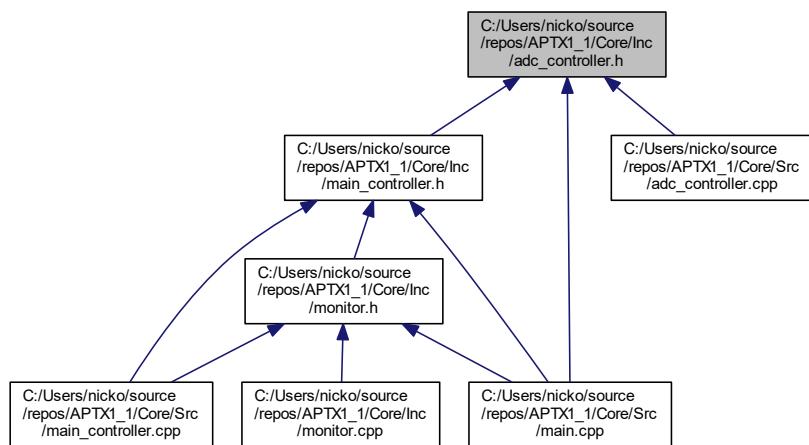
Definition at line 27 of file adc.c.

6.2 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc_controller.h File Reference

```
#include "main.h"
#include <stdint.h>
#include "cs_register.h"
Include dependency graph for adc_controller.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- union `adc_command`
- union `channel`
- class `AdcController`

Enumerations

- enum `select_channel` {
 Pressure, Motor_current_sense, V_Motor, V12,
 V5, VDD_sense, Temp, Undefined }

6.2.1 Enumeration Type Documentation

6.2.1.1 `select_channel`

enum `select_channel`

Enumerator

Pressure	
Motor_current_sense	
V_Motor	
V12	
V5	
VDD_sense	
Temp	
Undefined	

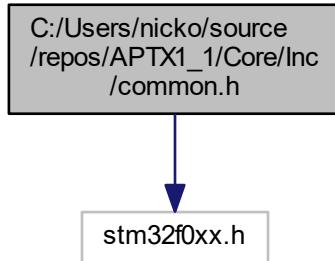
Definition at line 74 of file adc_controller.h.

```
75 {
76     Pressure,
77     Motor_current_sense,
78     V_Motor,
79     V12,
80     V5,
81     VDD_sense,
82     Temp,
83     Undefined
84 };
```

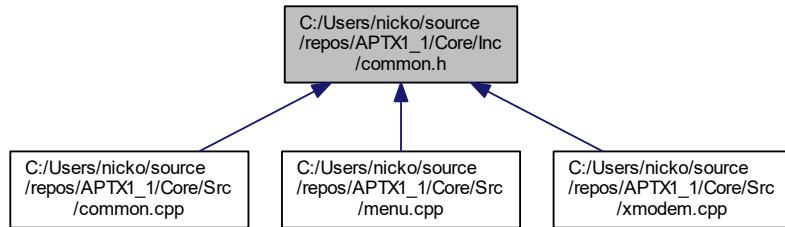
6.3 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/common.h File Reference

This file provides all the headers of the common functions.

```
#include "stm32f0xx.h"
Include dependency graph for common.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define TX_TIMEOUT ((uint32_t)100)
- #define RX_TIMEOUT HAL_MAX_DELAY
- #define IS_CAP LETTER(c) (((c) >= 'A') && ((c) <= 'F'))
- #define IS_LC LETTER(c) (((c) >= 'a') && ((c) <= 'f'))
- #define IS_09(c) (((c) >= '0') && ((c) <= '9'))
- #define ISVALIDHEX(c) (IS_CAP LETTER(c) || IS_LC LETTER(c) || IS_09(c))
- #define ISVALIDDEC(c) IS_09(c)
- #define CONVERTDEC(c) (c - '0')
- #define CONVERTHEX_ALPHA(c) (IS_CAP LETTER(c) ? ((c) - 'A'+10) : ((c) - 'a'+10))
- #define CONVERTHEX(c) (IS_09(c) ? ((c) - '0') : CONVERTHEX_ALPHA(c))

Functions

- void **Int2Str** (uint8_t *p_str, uint32_t intnum)

Convert an Integer to a string.
- uint32_t **Str2Int** (uint8_t *inputstr, uint32_t *intnum)

- Convert a string to an integer.*
- void **Serial_PutString** (const char *p_string)
- Print a string on the HyperTerminal.*
- HAL_StatusTypeDef **Serial_PutByte** (uint8_t param)
- Transmit a byte to the HyperTerminal.*

6.3.1 Detailed Description

This file provides all the headers of the common functions.

Author

MCD Application Team

Version

1.0.0

Date

8-April-2015

Attention

© COPYRIGHT(c) 2015 STMicroelectronics

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.3.2 Macro Definition Documentation

6.3.2.1 CONVERTDEC

```
#define CONVERTDEC(  
    c ) (c - '0')
```

Definition at line 57 of file common.h.

6.3.2.2 CONVERTHEX

```
#define CONVERTHEX(  
    c ) (IS_09(c) ? ((c) - '0') : CONVERTHEX_ALPHA(c))
```

Definition at line 60 of file common.h.

6.3.2.3 CONVERTHEX_ALPHA

```
#define CONVERTHEX_ALPHA(  
    c ) (IS_CAP_LETTER(c) ? ((c) - 'A'+10) : ((c) - 'a'+10))
```

Definition at line 59 of file common.h.

6.3.2.4 IS_09

```
#define IS_09(  
    c ) (((c) >= '0') && ((c) <= '9'))
```

Definition at line 54 of file common.h.

6.3.2.5 IS_CAP_LETTER

```
#define IS_CAP_LETTER(  
    c ) (((c) >= 'A') && ((c) <= 'F'))
```

Definition at line 52 of file common.h.

6.3.2.6 IS_LC LETTER

```
#define IS_LC LETTER( c ) (((c) >= 'a') && ((c) <= 'f'))
```

Definition at line 53 of file common.h.

6.3.2.7 ISVALIDDEC

```
#define ISVALIDDEC( c ) IS_09(c)
```

Definition at line 56 of file common.h.

6.3.2.8 ISVALIDHEX

```
#define ISVALIDHEX( c ) (IS_CAP LETTER(c) || IS_LC LETTER(c) || IS_09(c))
```

Definition at line 55 of file common.h.

6.3.2.9 RX_TIMEOUT

```
#define RX_TIMEOUT HAL_MAX_DELAY
```

Definition at line 49 of file common.h.

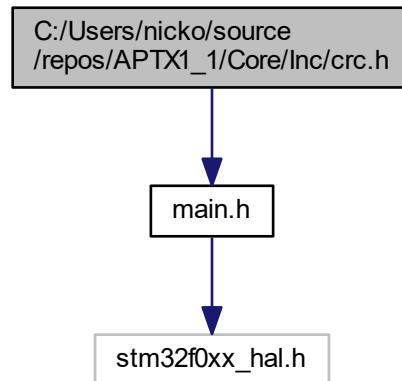
6.3.2.10 TX_TIMEOUT

```
#define TX_TIMEOUT ((uint32_t)100)
```

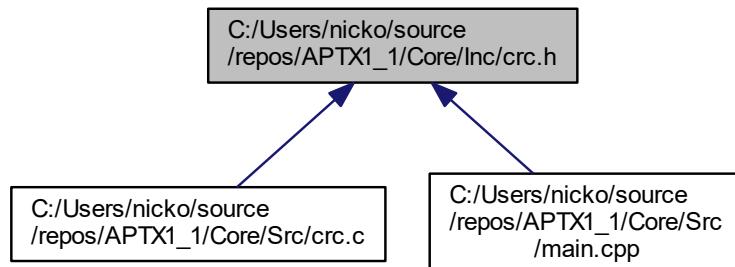
Definition at line 48 of file common.h.

6.4 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/crc.h File Reference

```
#include "main.h"  
Include dependency graph for crc.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_CRC_Init](#) (void)

Variables

- CRC_HandleTypeDef [hcrc](#)

6.4.1 Function Documentation

6.4.1.1 MX_CRC_Init()

```
void MX_CRC_Init (
    void )
```

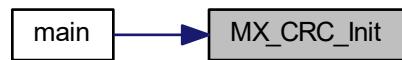
Definition at line 30 of file crc.c.

```
31 {
32
33     hcrc.Instance = CRC;
34     hcrc.Init.DefaultPolynomialUse = DEFAULT_POLYNOMIAL_ENABLE;
35     hcrc.Init.DefaultInitValueUse = DEFAULT_INIT_VALUE_ENABLE;
36     hcrc.Init.InputDataInversionMode = CRC_INPUTDATA_INVERSION_NONE;
37     hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
38     hcrc.InputDateFormat = CRC_INPUTDATA_FORMAT_BYTES;
39     if (HAL_CRC_Init(&hcrc) != HAL_OK)
40     {
41         Error_Handler();
42     }
43
44 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.4.2 Variable Documentation

6.4.2.1 hcrc

```
CRC_HandleTypeDef hcrc
```

File Name : [CRC.h](#) Description : This file provides code for the configuration of the CRC instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [CRC.c](#) Description : This file provides code for the configuration of the CRC instances.

Attention

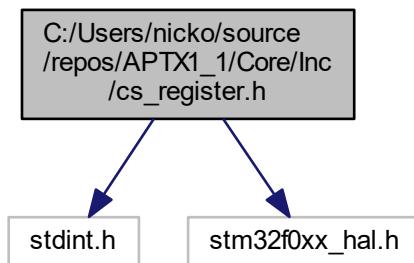
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

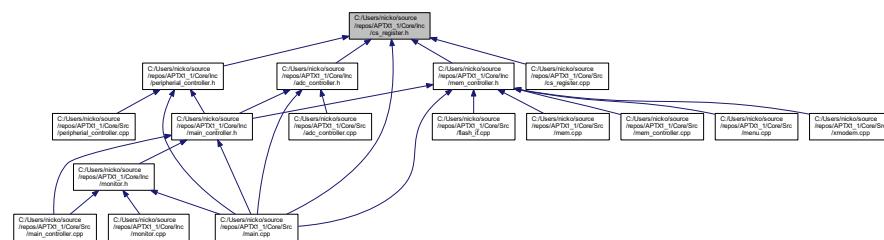
Definition at line 27 of file crc.c.

6.5 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h File Reference

```
#include <stdint.h>
#include "stm32f0xx_hal.h"
Include dependency graph for cs_register.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- union [byte8_t_reg_cs](#)
- union [byte8_t_reg_readback](#)
- class [CS_reg](#)

Enumerations

- enum [cs_decoder](#) {
 [Y0](#), [OPTO_CNT_CLRN](#), [CS_A2D](#), [CS_STATUS](#),
[CS_MEM3](#), [CS_MEM2](#), [CS_READBACK_SER](#) }

6.5.1 Enumeration Type Documentation

6.5.1.1 [cs_decoder](#)

enum [cs_decoder](#)

Enumerator

Y0	
OPTO_CNT_CLRN	
CS_A2D	
CS_STATUS	
CS_MEM3	
CS_MEM2	
CS_READBACK_SER	

Definition at line 46 of file [cs_register.h](#).

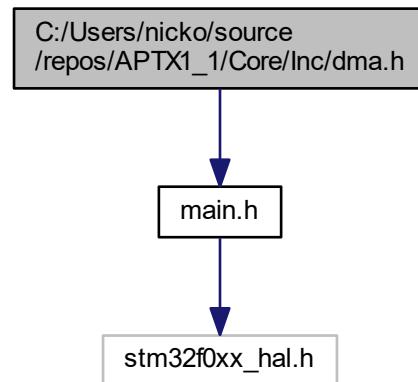
```

47 {
48     Y0,
49     OPTO\_CNT\_CLRN,
50     CS\_A2D,
51     CS\_STATUS,
52     CS\_MEM3,
53     CS\_MEM2,
54     CS\_READBACK\_SER
55 };

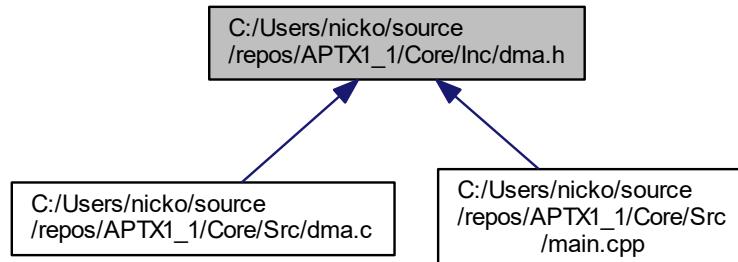
```

6.6 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/dma.h File Reference

```
#include "main.h"
Include dependency graph for dma.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_DMA_Init](#) (void)

6.6.1 Function Documentation

6.6.1.1 MX_DMA_Init()

```
void MX_DMA_Init (
    void )
```

File Name : [dma.h](#) Description : This file contains all the function prototypes for the [dma.c](#) file

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [dma.c](#) Description : This file provides code for the configuration of all the requested memory to memory DMA transfers.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause Enable DMA controller clock

Definition at line 38 of file [dma.c](#).

```
39 {
40
41     /* DMA controller clock enable */
42     __HAL_RCC_DMA1_CLK_ENABLE();
43
44     /* DMA interrupt init */
45     /* DMA1_Channel1_IRQHandler interrupt configuration */
46     HAL_NVIC_SetPriority(DMA1_Channel1_IRQn, 0, 0);
47     HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQn);
48
49 }
```

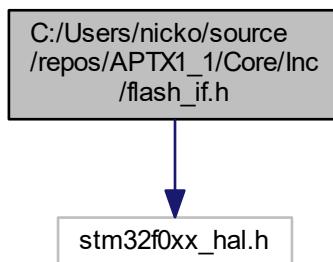
Here is the caller graph for this function:



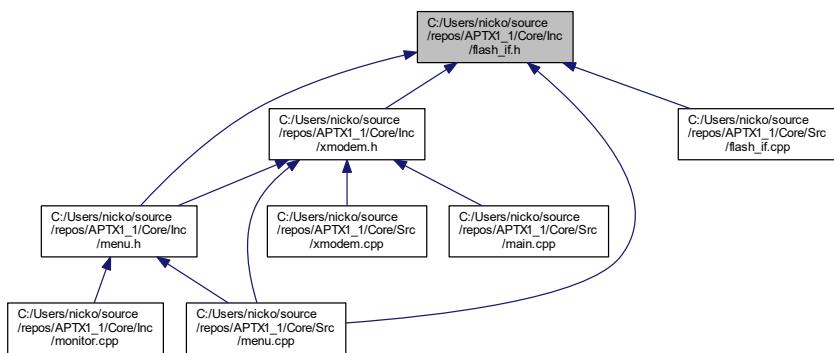
6.7 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/flash_if.h File Reference

This file provides all the headers of the flash_if functions.

```
#include "stm32f0xx_hal.h"
Include dependency graph for flash_if.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define **FLASH_PAGE_STEP** 0x100 /* Size of page : 256 bytes */
- #define **APPLICATION_ADDRESS** (uint32_t)0x00000000 /* Start user code address: ADDR_FLASH_PA←GE_8 */
- #define **USER_FLASH_SIZE** 0x100000
- #define **ABS_RETURN**(x, y) (((x) < (y)) ? (y) : (x))

Enumerations

- enum {
 `FLASHIF_OK` = 0, `FLASHIF_ERASEKO`, `FLASHIF_WRITINGCTRL_ERROR`, `FLASHIF_WRITING_ERROR`,
`FLASHIF_PROTECTION_ERRROR` }
- enum { `FLASHIF_PROTECTION_NONE` = 0, `FLASHIF_PROTECTION_PCROENABLED` = 0x1,
`FLASHIF_PROTECTION_WRPENABLED` = 0x2, `FLASHIF_PROTECTION_RDPENABLED` = 0x4 }
- enum { `FLASHIF_WRP_ENABLE`, `FLASHIF_WRP_DISABLE` }

Functions

- void `FLASH_If_Erase` (void)
This function does an erase of all user flash area.
- uint32_t `FLASH_If_Write` (uint32_t destination, uint8_t *p_source, uint32_t length)
This function writes a data buffer in flash (data are 32-bit aligned).

6.7.1 Detailed Description

This file provides all the headers of the flash_if functions.

Author

MCD Application Team

Version

1.0.0

Date

8-April-2015

Attention

© COPYRIGHT(c) 2015 STMicroelectronics

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.7.2 Macro Definition Documentation

6.7.2.1 ABS_RETURN

```
#define ABS_RETURN(  
    x,  
    y ) (( (x) < (y) ) ? (y) : (x))
```

Definition at line 82 of file flash_if.h.

6.7.2.2 APPLICATION_ADDRESS

```
#define APPLICATION_ADDRESS (uint32_t)0x00000000 /* Start user code address: ADDR_FLASH_PAGE←  
_8 */
```

Definition at line 76 of file flash_if.h.

6.7.2.3 FLASH_PAGE_STEP

```
#define FLASH_PAGE_STEP 0x100 /* Size of page : 256 bytes */
```

Definition at line 75 of file flash_if.h.

6.7.2.4 USER_FLASH_SIZE

```
#define USER_FLASH_SIZE 0x100000
```

Definition at line 77 of file flash_if.h.

6.7.3 Enumeration Type Documentation

6.7.3.1 anonymous enum

```
anonymous enum
```

Enumerator

FLASHIF_OK	
FLASHIF_ERASEKO	
FLASHIF_WRITINGCTRL_ERROR	
FLASHIF_WRITING_ERROR	
FLASHIF_PROTECTION_ERROR	

Definition at line 50 of file flash_if.h.

```
51 {
52     FLASHIF_OK = 0,
53     FLASHIF_ERASEKO,
54     FLASHIF_WRITINGCTRL_ERROR,
55     FLASHIF_WRITING_ERROR,
56     FLASHIF_PROTECTION_ERROR
57 };
```

6.7.3.2 anonymous enum

anonymous enum

Enumerator

FLASHIF_PROTECTION_NONE	
FLASHIF_PROTECTION_PCROPENABLED	
FLASHIF_PROTECTION_WRPENABLED	
FLASHIF_PROTECTION_RDPENABLED	

Definition at line 60 of file flash_if.h.

```
60     {
61     FLASHIF_PROTECTION_NONE      = 0,
62     FLASHIF_PROTECTION_PCROPENABLED = 0x1,
63     FLASHIF_PROTECTION_WRPENABLED   = 0x2,
64     FLASHIF_PROTECTION_RDPENABLED   = 0x4,
65 };
```

6.7.3.3 anonymous enum

anonymous enum

Enumerator

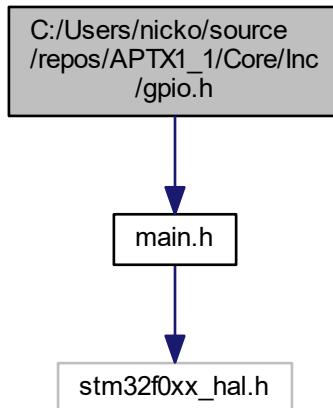
FLASHIF_WRP_ENABLE	
FLASHIF_WRP_DISABLE	

Definition at line 68 of file flash_if.h.

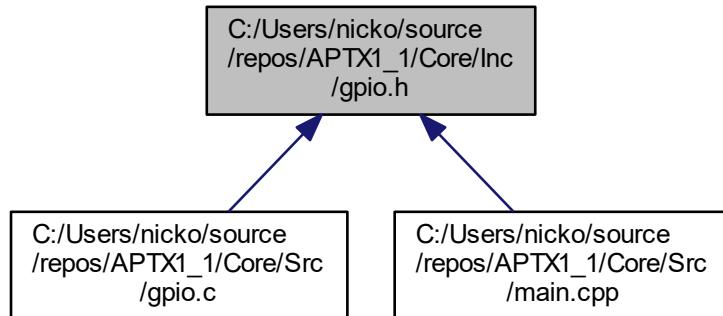
```
68     {
69     FLASHIF_WRP_ENABLE,
70     FLASHIF_WRP_DISABLE
71 };
```

6.8 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/gpio.h File Reference

```
#include "main.h"  
Include dependency graph for gpio.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_GPIO_Init](#) (void)

6.8.1 Function Documentation

6.8.1.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
    void )
```

File Name : [gpio.h](#) Description : This file contains all the functions prototypes for the gpio

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [gpio.c](#) Description : This file provides code for the configuration of all used GPIO pins.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

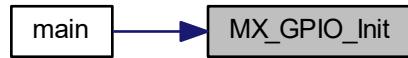
This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause Configure pins as Analog Input Output EVENT_OUT EXTI PA8 ----> RCC_MCO

Definition at line 41 of file gpio.c.

```
42 {
43
44     GPIO_InitTypeDef GPIO_InitStruct = {0};
45
46     /* GPIO Ports Clock Enable */
47     __HAL_RCC_GPIOC_CLK_ENABLE();
48     __HAL_RCC_GPIOF_CLK_ENABLE();
49     __HAL_RCC_GPIOA_CLK_ENABLE();
50     __HAL_RCC_GPIOB_CLK_ENABLE();
51     __HAL_RCC_GPIOD_CLK_ENABLE();
52
53     /*Configure GPIO pin Output Level */
54     HAL_GPIO_WritePin(GPIOC,
55                         ENABLE_POWER_12V_Pin|LVDS_RESERVED_Pin|CHIP_SELECT_RESERVED_Pin|RESERVED_UART6_TX_Pin
56                         |RESERVED_UART6_RX_Pin|DEBUG_MOTOR_RUNNING_Pin|DEBUG_PRESSURE_OK_Pin,
57                         GPIO_PIN_RESET);
58
59     /*Configure GPIO pin Output Level */
60     HAL_GPIO_WritePin(GPIOA, CHIP_SELECT_SERIAL_Pin|USB_DM_RESERVED_Pin|USB_DP_RESERVED_Pin,
61                         GPIO_PIN_RESET);
62
63     /*Configure GPIO pin Output Level */
64     HAL_GPIO_WritePin(GPIOB, CHIP_SELECT MCU_MEM_Pin|WATCHDOG_OUT_Pin|DEBUG_CLK_AUX_Pin|BUZZER_Pin
65                         |BLINKING_LED_Pin|DEBUG_AM_OK_Pin|DEBUG_APT_OK_Pin|DEBUG MCU_OK_Pin
66                         |SYS_SWO_RESERVED_Pin|DEBUG_DATA_AUX_Pin|DEBUG_STATE_3_Pin|DEBUG_STATE_2_Pin
67                         |DEBUG_STATE_1_Pin|DEBUG_BRK1_Pin|DEBUG_BRK2_Pin, GPIO_PIN_RESET);
```

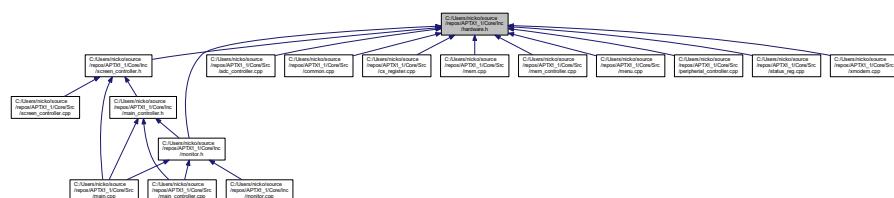
```
65  /*Configure GPIO pin Output Level */
66  HAL_GPIO_WritePin(DEBUG_LATCH_AUX_GPIO_Port, DEBUG_LATCH_AUX_Pin, GPIO_PIN_RESET);
67
68  /*Configure GPIO pins : PCPin PCPin PCPin PCPin
69   *                      PCPin */
70  GPIO_InitStruct.Pin =
71    TOGGLE400_RESET_BTN_Pin|SIMPLE_INTERLOCK_Pin|LOGIC_BYPASS_3_Pin|LOGIC_BYPASS_2_Pin
72    |LOGIC_BYPASS_1_Pin;
73  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
74  GPIO_InitStruct.Pull = GPIO_NOPULL;
75  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
76
77  /*Configure GPIO pins : PCPin PCPin PCPin PCPin
78   *                      PCPin PCPin PCPin */
79  GPIO_InitStruct.Pin =
80    ENABLE_POWER_12V_Pin|LVDS_RESERVED_Pin|CHIP_SELECT_RESERVED_Pin|RESERVED_UART6_TX_Pin
81    |RESERVED_UART6_RX_Pin|DEBUG_MOTOR_RUNNING_Pin|DEBUG_PRESSURE_OK_Pin;
82  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
83  GPIO_InitStruct.Pull = GPIO_NOPULL;
84  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
85  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
86
87  /*Configure GPIO pins : PAPin PAPin */
88  GPIO_InitStruct.Pin = PA1_RESERVED_ASK_SHAUL_Pin|LOGIC_BYPASS_4_Pin;
89  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
90  GPIO_InitStruct.Pull = GPIO_NOPULL;
91  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
92
93  /*Configure GPIO pins : PAPin PAPin PAPin */
94  GPIO_InitStruct.Pin = CHIP_SELECT_SERIAL_Pin|USB_DM_RESERVED_Pin|USB_DP_RESERVED_Pin;
95  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
96  GPIO_InitStruct.Pull = GPIO_NOPULL;
97  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
98  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
99
100  /*Configure GPIO pins : PBPin PBPin PBPin PBPin
101   *                      PBPin PBPin PBPin PBPin
102   *                      PBPin PBPin PBPin */
103  GPIO_InitStruct.Pin = CHIP_SELECT MCU_MEM_Pin|WATCHDOG_OUT_Pin|DEBUG_CLK_AUX_Pin|BUZZER_Pin
104    |BLINKING_LED_Pin|DEBUG_AM_OK_Pin|DEBUG_APT_OK_Pin|DEBUG MCU_OK_Pin
105    |SYS_SWO_RESERVED_Pin|DEBUG_DATA_AUX_Pin|DEBUG_STATE_3_Pin|DEBUG_STATE_2_Pin
106    |DEBUG_STATE_1_Pin|DEBUG_BRK1_Pin|DEBUG_BRK2_Pin;
107  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
108  GPIO_InitStruct.Pull = GPIO_NOPULL;
109  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
110  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
111
112  /*Configure GPIO pin : PtPin */
113  GPIO_InitStruct.Pin = POWER_FAIL MCU_Pin;
114  GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
115  GPIO_InitStruct.Pull = GPIO_NOPULL;
116  HAL_GPIO_Init(POWER_FAIL MCU GPIO_Port, &GPIO_InitStruct);
117
118  /*Configure GPIO pin : PA8 */
119  GPIO_InitStruct.Pin = GPIO_PIN_8;
120  GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
121  GPIO_InitStruct.Pull = GPIO_NOPULL;
122  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
123  GPIO_InitStruct.Alternate = GPIO_AF0_MCO;
124  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
125
126  /*Configure GPIO pin : PtPin */
127  GPIO_InitStruct.Pin = DEBUG_LATCH_AUX_Pin;
128  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
129  GPIO_InitStruct.Pull = GPIO_NOPULL;
130  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
131  HAL_GPIO_Init(DEBUG_LATCH_AUX_GPIO_Port, &GPIO_InitStruct);
132
133 }
```

Here is the caller graph for this function:



6.9 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/hardware.h File Reference

This graph shows which files directly or indirectly include this file:



Macros

- #define DEBUG_PRINT 0
- #define TECHNICIAN_ENABLED 0
- #define PASS_SPI 0
- #define PASS_PRESSURE_CHECK 0
- #define EXPERIMENTAL 0
- #define DONT_BEEP 0
- #define FROM_BOOTLOADER 1
- #define MAX_HOLE_SIZE 5
- #define ADC_Presure 0
- #define ADC_Battery 1
- #define ADC_Presure_27_bar 500
- #define ADC_Presure_20_bar 400
- #define VERSION "7.3"
- #define MCU_VERSION_MAJOR 7
- #define MCU_VERSION_MINOR 3
- #define MCU_VERSION_PATCH 0
- #define MCU_VERSION_RC 1
- #define APP_ADDRESS (uint32_t)0x08008000
- #define APP_ADDRESS_P (uint32_t*)APP_ADDRESS
- #define END_ADDRESS (uint32_t)0x08020000
- #define APP_SIZE (uint32_t)(END_ADDRESS - APP_ADDRESS)
- #define READ_ARRAY1 0x0B
- #define READ_ARRAY2 0x03

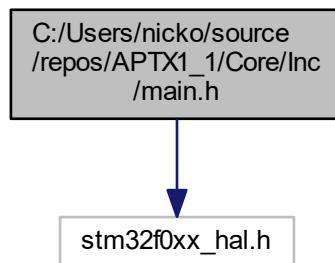
- #define READ_DUAL 0x3B
- #define READ_QUAD 0x6B
- #define BLOCK_ERASE_4K 0x20
- #define BLOCK_ERASE_32K 0x52
- #define BLOCK_ERASE_64K 0xD8
- #define ERASE_CHIP1 0x60
- #define ERASE_CHIP2 0xC7
- #define BYTE_PROGRAM 0x02
- #define WRITE_ENABLE 0x06
- #define WRITE_DISABLE 0x04
- #define READ_STATUS_REGISTER1 0x05
- #define READ_STATUS_REGISTER2 0x35
- #define WRITE_STATUS_REGISTER 0x01
- #define WRITE_ENABLE_STATUS_REGISTER_VOLATILE 0x50
- #define READ_MANUFACTURE_ID 0x9F
- #define READ_DEVICE_ID 0x90
- #define HIGH (GPIO_PinState)1
- #define LOW (GPIO_PinState)0
- #define HDC_1080_ADD 0x40
- #define Configuration_register_add 0x02
- #define Temperature_register_add 0x00
- #define Humidity_register_add 0x01
- #define START_BEEP 100
- #define BEEP_COUNTER_200 110
- #define BEEP_COUNTER_400 120
- #define BEEP_COUNTER_0 130
- #define KEY_PRESS 80
- #define Beep_Every_X_Counting 200
- #define SERIAL_PC &huart1
- #define SERIAL_SCREEN &huart2
- #define RXBUFSIZE 256
- #define TIMEOUT_SCREEN 5000
- #define TICKS_FOR_UPDATE 500
- #define TICKS_SCREEN_UPDATE_LONG 1250
- #define TICKS_SCREEN_UPDATE_SUPER_LONG 3500
- #define AM_WARNING_TIMES 3
- #define MOTOR_1_PWM_ON_50 50
- #define MOTOR_1_PWM_ON_75 75
- #define MOTOR_1_PWM_ON 100
- #define MOTOR_1_PWM_OFF 0
- #define SELECT_CHANNEL_0 0b0000
- #define SELECT_CHANNEL_1 0b0001
- #define SELECT_CHANNEL_2 0b0010
- #define SELECT_CHANNEL_3 0b0011
- #define SELECT_CHANNEL_4 0b0100
- #define SELECT_CHANNEL_5 0b0101
- #define SELECT_CHANNEL_6 0b0110
- #define SELECT_CHANNEL_7 0b0111
- #define ADC_RESERVED 0b1000
- #define ADC_RESERVED2 0b1001
- #define WRITE_CFR 0b1010
- #define SELECT_TEST_VDIV2 0b1011
- #define SELECT_TEST_REFM 0b1100
- #define SELECT_TEST_REFP 0b1101
- #define FIFOREAD 0b1110
- #define HARDWARE_DEFAULT 0b1111

6.10 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main.h File Reference

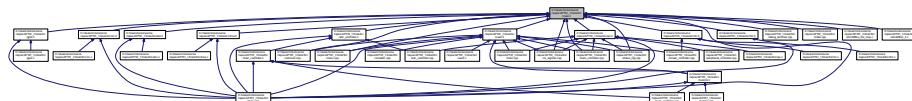
: Header for main.c file. This file contains the common defines of the application.

```
#include "stm32f0xx_hal.h"
```

Include dependency graph for main.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define RTC_TIMESTAMP_Pin GPIO_PIN_13
- #define RTC_TIMESTAMP_GPIO_Port GPIOC
- #define ADC_CURRENT_SENSE_Pin GPIO_PIN_0
- #define ADC_CURRENT_SENSE_GPIO_Port GPIOC
- #define TOGGLE400_RESET_BTN_Pin GPIO_PIN_1
- #define TOGGLE400_RESET_BTN_GPIO_Port GPIOC
- #define ENABLE_POWER_12V_Pin GPIO_PIN_2
- #define ENABLE_POWER_12V_GPIO_Port GPIOC
- #define SIMPLE_INTERLOCK_Pin GPIO_PIN_3
- #define SIMPLE_INTERLOCK_GPIO_Port GPIOC
- #define PA1_RESERVED_ASK_SHAUL_Pin GPIO_PIN_1
- #define PA1_RESERVED_ASK_SHAUL_GPIO_Port GPIOA
- #define UART2_SCREEN_TX_Pin GPIO_PIN_2
- #define UART2_SCREEN_TX_GPIO_Port GPIOA
- #define UART2_SCREEN_RX_Pin GPIO_PIN_3
- #define UART2_SCREEN_RX_GPIO_Port GPIOA
- #define CHIP_SELECT_SERIAL_Pin GPIO_PIN_4
- #define CHIP_SELECT_SERIAL_GPIO_Port GPIOA
- #define LVDS_RESERVED_Pin GPIO_PIN_4

- #define LVDS_RESERVED_GPIO_Port GPIOC
- #define CHIP_SELECT_RESERVED_Pin GPIO_PIN_5
- #define CHIP_SELECT_RESERVED_GPIO_Port GPIOC
- #define CHIP_SELECT MCU_MEM_Pin GPIO_PIN_0
- #define CHIP_SELECT MCU_MEM_GPIO_Port GPIOB
- #define WATCHDOG_OUT_Pin GPIO_PIN_1
- #define WATCHDOG_OUT_GPIO_Port GPIOB
- #define POWER_FAIL MCU_Pin GPIO_PIN_2
- #define POWER_FAIL MCU_GPIO_Port GPIOB
- #define DEBUG_CLK_AUX_Pin GPIO_PIN_10
- #define DEBUG_CLK_AUX_GPIO_Port GPIOB
- #define BUZZER_Pin GPIO_PIN_11
- #define BUZZER_GPIO_Port GPIOB
- #define BLINKING_LED_Pin GPIO_PIN_12
- #define BLINKING_LED_GPIO_Port GPIOB
- #define DEBUG_AM_OK_Pin GPIO_PIN_13
- #define DEBUG_AM_OK_GPIO_Port GPIOB
- #define DEBUG_APT_OK_Pin GPIO_PIN_14
- #define DEBUG_APT_OK_GPIO_Port GPIOB
- #define DEBUG_MCU_OK_Pin GPIO_PIN_15
- #define DEBUG_MCU_OK_GPIO_Port GPIOB
- #define RESERVED_UART6_TX_Pin GPIO_PIN_6
- #define RESERVED_UART6_TX_GPIO_Port GPIOC
- #define RESERVED_UART6_RX_Pin GPIO_PIN_7
- #define RESERVED_UART6_RX_GPIO_Port GPIOC
- #define DEBUG_MOTOR_RUNNING_Pin GPIO_PIN_8
- #define DEBUG_MOTOR_RUNNING_GPIO_Port GPIOC
- #define DEBUG_PRESSURE_OK_Pin GPIO_PIN_9
- #define DEBUG_PRESSURE_OK_GPIO_Port GPIOC
- #define UART1_PC_TX_Pin GPIO_PIN_9
- #define UART1_PC_TX_GPIO_Port GPIOA
- #define UART1_PC_RX_Pin GPIO_PIN_10
- #define UART1_PC_RX_GPIO_Port GPIOA
- #define USB_DM_RESERVED_Pin GPIO_PIN_11
- #define USB_DM_RESERVED_GPIO_Port GPIOA
- #define USB_DP_RESERVED_Pin GPIO_PIN_12
- #define USB_DP_RESERVED_GPIO_Port GPIOA
- #define LOGIC_BYPASS_4_Pin GPIO_PIN_15
- #define LOGIC_BYPASS_4_GPIO_Port GPIOA
- #define LOGIC_BYPASS_3_Pin GPIO_PIN_10
- #define LOGIC_BYPASS_3_GPIO_Port GPIOC
- #define LOGIC_BYPASS_2_Pin GPIO_PIN_11
- #define LOGIC_BYPASS_2_GPIO_Port GPIOC
- #define LOGIC_BYPASS_1_Pin GPIO_PIN_12
- #define LOGIC_BYPASS_1_GPIO_Port GPIOC
- #define DEBUG_LATCH_AUX_Pin GPIO_PIN_2
- #define DEBUG_LATCH_AUX_GPIO_Port GPIOD
- #define SYS_SWO_RESERVED_Pin GPIO_PIN_3
- #define SYS_SWO_RESERVED_GPIO_Port GPIOB
- #define DEBUG_DATA_AUX_Pin GPIO_PIN_4
- #define DEBUG_DATA_AUX_GPIO_Port GPIOB
- #define DEBUG_STATE_3_Pin GPIO_PIN_5
- #define DEBUG_STATE_3_GPIO_Port GPIOB
- #define DEBUG_STATE_2_Pin GPIO_PIN_6
- #define DEBUG_STATE_2_GPIO_Port GPIOB

- #define DEBUG_STATE_1_Pin GPIO_PIN_7
- #define DEBUG_STATE_1_GPIO_Port GPIOB
- #define DEBUG_BRK1_Pin GPIO_PIN_8
- #define DEBUG_BRK1_GPIO_Port GPIOB
- #define DEBUG_BRK2_Pin GPIO_PIN_9
- #define DEBUG_BRK2_GPIO_Port GPIOB
- #define DMA_LENGTH 4

Functions

- void **Error_Handler** (void)
This function is executed in case of error occurrence.

6.10.1 Detailed Description

: Header for main.c file. This file contains the common defines of the application.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

6.10.2 Macro Definition Documentation

6.10.2.1 ADC_CURRENT_SENSE_GPIO_Port

```
#define ADC_CURRENT_SENSE_GPIO_Port GPIOC
```

Definition at line 64 of file main.h.

6.10.2.2 ADC_CURRENT_SENSE_Pin

```
#define ADC_CURRENT_SENSE_Pin GPIO_PIN_0
```

Definition at line 63 of file main.h.

6.10.2.3 BLINKING_LED_GPIO_Port

```
#define BLINKING_LED_GPIO_Port GPIOB
```

Definition at line 94 of file main.h.

6.10.2.4 BLINKING_LED_Pin

```
#define BLINKING_LED_Pin GPIO_PIN_12
```

Definition at line 93 of file main.h.

6.10.2.5 BUZZER_GPIO_Port

```
#define BUZZER_GPIO_Port GPIOB
```

Definition at line 92 of file main.h.

6.10.2.6 BUZZER_Pin

```
#define BUZZER_Pin GPIO_PIN_11
```

Definition at line 91 of file main.h.

6.10.2.7 CHIP_SELECT MCU MEM GPIO_Port

```
#define CHIP_SELECT_MCU_MEM_GPIO_Port GPIOB
```

Definition at line 84 of file main.h.

6.10.2.8 CHIP_SELECT MCU MEM Pin

```
#define CHIP_SELECT_MCU_MEM_Pin GPIO_PIN_0
```

Definition at line 83 of file main.h.

6.10.2.9 CHIP_SELECT_RESERVED_GPIO_Port

```
#define CHIP_SELECT_RESERVED_GPIO_Port GPIOC
```

Definition at line 82 of file main.h.

6.10.2.10 CHIP_SELECT_RESERVED_Pin

```
#define CHIP_SELECT_RESERVED_Pin GPIO_PIN_5
```

Definition at line 81 of file main.h.

6.10.2.11 CHIP_SELECT_SERIAL_GPIO_Port

```
#define CHIP_SELECT_SERIAL_GPIO_Port GPIOA
```

Definition at line 78 of file main.h.

6.10.2.12 CHIP_SELECT_SERIAL_Pin

```
#define CHIP_SELECT_SERIAL_Pin GPIO_PIN_4
```

Definition at line 77 of file main.h.

6.10.2.13 DEBUG_AM_OK_GPIO_Port

```
#define DEBUG_AM_OK_GPIO_Port GPIOB
```

Definition at line 96 of file main.h.

6.10.2.14 DEBUG_AM_OK_Pin

```
#define DEBUG_AM_OK_Pin GPIO_PIN_13
```

Definition at line 95 of file main.h.

6.10.2.15 DEBUG_APT_OK_GPIO_Port

```
#define DEBUG_APT_OK_GPIO_Port GPIOB
```

Definition at line 98 of file main.h.

6.10.2.16 DEBUG_APT_OK_Pin

```
#define DEBUG_APT_OK_Pin GPIO_PIN_14
```

Definition at line 97 of file main.h.

6.10.2.17 DEBUG_BRK1_GPIO_Port

```
#define DEBUG_BRK1_GPIO_Port GPIOB
```

Definition at line 138 of file main.h.

6.10.2.18 DEBUG_BRK1_Pin

```
#define DEBUG_BRK1_Pin GPIO_PIN_8
```

Definition at line 137 of file main.h.

6.10.2.19 DEBUG_BRK2_GPIO_Port

```
#define DEBUG_BRK2_GPIO_Port GPIOB
```

Definition at line 140 of file main.h.

6.10.2.20 DEBUG_BRK2_Pin

```
#define DEBUG_BRK2_Pin GPIO_PIN_9
```

Definition at line 139 of file main.h.

6.10.2.21 DEBUG_CLK_AUX_GPIO_Port

```
#define DEBUG_CLK_AUX_GPIO_Port GPIOB
```

Definition at line 90 of file main.h.

6.10.2.22 DEBUG_CLK_AUX_Pin

```
#define DEBUG_CLK_AUX_Pin GPIO_PIN_10
```

Definition at line 89 of file main.h.

6.10.2.23 DEBUG_DATA_AUX_GPIO_Port

```
#define DEBUG_DATA_AUX_GPIO_Port GPIOB
```

Definition at line 130 of file main.h.

6.10.2.24 DEBUG_DATA_AUX_Pin

```
#define DEBUG_DATA_AUX_Pin GPIO_PIN_4
```

Definition at line 129 of file main.h.

6.10.2.25 DEBUG_LATCH_AUX_GPIO_Port

```
#define DEBUG_LATCH_AUX_GPIO_Port GPIOD
```

Definition at line 126 of file main.h.

6.10.2.26 DEBUG_LATCH_AUX_Pin

```
#define DEBUG_LATCH_AUX_Pin GPIO_PIN_2
```

Definition at line 125 of file main.h.

6.10.2.27 DEBUG MCU OK GPIO Port

```
#define DEBUG MCU OK GPIO Port GPIOB
```

Definition at line 100 of file main.h.

6.10.2.28 DEBUG MCU OK Pin

```
#define DEBUG MCU OK Pin GPIO_PIN_15
```

Definition at line 99 of file main.h.

6.10.2.29 DEBUG MOTOR RUNNING GPIO Port

```
#define DEBUG MOTOR RUNNING GPIO Port GPIOC
```

Definition at line 106 of file main.h.

6.10.2.30 DEBUG MOTOR RUNNING Pin

```
#define DEBUG MOTOR RUNNING Pin GPIO_PIN_8
```

Definition at line 105 of file main.h.

6.10.2.31 DEBUG PRESSURE OK GPIO Port

```
#define DEBUG PRESSURE OK GPIO Port GPIOC
```

Definition at line 108 of file main.h.

6.10.2.32 DEBUG PRESSURE OK Pin

```
#define DEBUG PRESSURE OK Pin GPIO_PIN_9
```

Definition at line 107 of file main.h.

6.10.2.33 DEBUG_STATE_1_GPIO_Port

```
#define DEBUG_STATE_1_GPIO_Port GPIOB
```

Definition at line 136 of file main.h.

6.10.2.34 DEBUG_STATE_1_Pin

```
#define DEBUG_STATE_1_Pin GPIO_PIN_7
```

Definition at line 135 of file main.h.

6.10.2.35 DEBUG_STATE_2_GPIO_Port

```
#define DEBUG_STATE_2_GPIO_Port GPIOB
```

Definition at line 134 of file main.h.

6.10.2.36 DEBUG_STATE_2_Pin

```
#define DEBUG_STATE_2_Pin GPIO_PIN_6
```

Definition at line 133 of file main.h.

6.10.2.37 DEBUG_STATE_3_GPIO_Port

```
#define DEBUG_STATE_3_GPIO_Port GPIOB
```

Definition at line 132 of file main.h.

6.10.2.38 DEBUG_STATE_3_Pin

```
#define DEBUG_STATE_3_Pin GPIO_PIN_5
```

Definition at line 131 of file main.h.

6.10.2.39 DMA_LENGTH

```
#define DMA_LENGTH 4
```

Definition at line 142 of file main.h.

6.10.2.40 ENABLE_POWER_12V_GPIO_Port

```
#define ENABLE_POWER_12V_GPIO_Port GPIOC
```

Definition at line 68 of file main.h.

6.10.2.41 ENABLE_POWER_12V_Pin

```
#define ENABLE_POWER_12V_Pin GPIO_PIN_2
```

Definition at line 67 of file main.h.

6.10.2.42 LOGIC_BYPASS_1_GPIO_Port

```
#define LOGIC_BYPASS_1_GPIO_Port GPIOC
```

Definition at line 124 of file main.h.

6.10.2.43 LOGIC_BYPASS_1_Pin

```
#define LOGIC_BYPASS_1_Pin GPIO_PIN_12
```

Definition at line 123 of file main.h.

6.10.2.44 LOGIC_BYPASS_2_GPIO_Port

```
#define LOGIC_BYPASS_2_GPIO_Port GPIOC
```

Definition at line 122 of file main.h.

6.10.2.45 LOGIC_BYPASS_2_Pin

```
#define LOGIC_BYPASS_2_Pin GPIO_PIN_11
```

Definition at line 121 of file main.h.

6.10.2.46 LOGIC_BYPASS_3_GPIO_Port

```
#define LOGIC_BYPASS_3_GPIO_Port GPIOC
```

Definition at line 120 of file main.h.

6.10.2.47 LOGIC_BYPASS_3_Pin

```
#define LOGIC_BYPASS_3_Pin GPIO_PIN_10
```

Definition at line 119 of file main.h.

6.10.2.48 LOGIC_BYPASS_4_GPIO_Port

```
#define LOGIC_BYPASS_4_GPIO_Port GPIOA
```

Definition at line 118 of file main.h.

6.10.2.49 LOGIC_BYPASS_4_Pin

```
#define LOGIC_BYPASS_4_Pin GPIO_PIN_15
```

Definition at line 117 of file main.h.

6.10.2.50 LVDS_RESERVED_GPIO_Port

```
#define LVDS_RESERVED_GPIO_Port GPIOC
```

Definition at line 80 of file main.h.

6.10.2.51 LVDS_RESERVED_Pin

```
#define LVDS_RESERVED_Pin GPIO_PIN_4
```

Definition at line 79 of file main.h.

6.10.2.52 PA1_RESERVED_ASK_SHAUL_GPIO_Port

```
#define PA1_RESERVED_ASK_SHAUL_GPIO_Port GPIOA
```

Definition at line 72 of file main.h.

6.10.2.53 PA1_RESERVED_ASK_SHAUL_Pin

```
#define PA1_RESERVED_ASK_SHAUL_Pin GPIO_PIN_1
```

Definition at line 71 of file main.h.

6.10.2.54 POWER_FAIL MCU GPIO_Port

```
#define POWER_FAIL MCU_GPIO_Port GPIOB
```

Definition at line 88 of file main.h.

6.10.2.55 POWER_FAIL MCU Pin

```
#define POWER_FAIL MCU_Pin GPIO_PIN_2
```

Definition at line 87 of file main.h.

6.10.2.56 RESERVED_UART6_RX_GPIO_Port

```
#define RESERVED_UART6_RX_GPIO_Port GPIOC
```

Definition at line 104 of file main.h.

6.10.2.57 RESERVED_UART6_RX_Pin

```
#define RESERVED_UART6_RX_Pin GPIO_PIN_7
```

Definition at line 103 of file main.h.

6.10.2.58 RESERVED_UART6_TX_GPIO_Port

```
#define RESERVED_UART6_TX_GPIO_Port GPIOC
```

Definition at line 102 of file main.h.

6.10.2.59 RESERVED_UART6_TX_Pin

```
#define RESERVED_UART6_TX_Pin GPIO_PIN_6
```

Definition at line 101 of file main.h.

6.10.2.60 RTC_TIMESTAMP_GPIO_Port

```
#define RTC_TIMESTAMP_GPIO_Port GPIOC
```

Definition at line 62 of file main.h.

6.10.2.61 RTC_TIMESTAMP_Pin

```
#define RTC_TIMESTAMP_Pin GPIO_PIN_13
```

Definition at line 61 of file main.h.

6.10.2.62 SIMPLE_INTERLOCK_GPIO_Port

```
#define SIMPLE_INTERLOCK_GPIO_Port GPIOC
```

Definition at line 70 of file main.h.

6.10.2.63 SIMPLE_INTERLOCK_Pin

```
#define SIMPLE_INTERLOCK_Pin GPIO_PIN_3
```

Definition at line 69 of file main.h.

6.10.2.64 SYS_SWO_RESERVED_GPIO_Port

```
#define SYS_SWO_RESERVED_GPIO_Port GPIOB
```

Definition at line 128 of file main.h.

6.10.2.65 SYS_SWO_RESERVED_Pin

```
#define SYS_SWO_RESERVED_Pin GPIO_PIN_3
```

Definition at line 127 of file main.h.

6.10.2.66 TOGGLE400_RESET_BTN_GPIO_Port

```
#define TOGGLE400_RESET_BTN_GPIO_Port GPIOC
```

Definition at line 66 of file main.h.

6.10.2.67 TOGGLE400_RESET_BTN_Pin

```
#define TOGGLE400_RESET_BTN_Pin GPIO_PIN_1
```

Definition at line 65 of file main.h.

6.10.2.68 UART1_PC_RX_GPIO_Port

```
#define UART1_PC_RX_GPIO_Port GPIOA
```

Definition at line 112 of file main.h.

6.10.2.69 **UART1_PC_RX_Pin**

```
#define UART1_PC_RX_Pin GPIO_PIN_10
```

Definition at line 111 of file main.h.

6.10.2.70 **UART1_PC_TX_GPIO_Port**

```
#define UART1_PC_TX_GPIO_Port GPIOA
```

Definition at line 110 of file main.h.

6.10.2.71 **UART1_PC_TX_Pin**

```
#define UART1_PC_TX_Pin GPIO_PIN_9
```

Definition at line 109 of file main.h.

6.10.2.72 **UART2_SCREEN_RX_GPIO_Port**

```
#define UART2_SCREEN_RX_GPIO_Port GPIOA
```

Definition at line 76 of file main.h.

6.10.2.73 **UART2_SCREEN_RX_Pin**

```
#define UART2_SCREEN_RX_Pin GPIO_PIN_3
```

Definition at line 75 of file main.h.

6.10.2.74 **UART2_SCREEN_TX_GPIO_Port**

```
#define UART2_SCREEN_TX_GPIO_Port GPIOA
```

Definition at line 74 of file main.h.

6.10.2.75 **UART2_SCREEN_TX_Pin**

```
#define UART2_SCREEN_TX_Pin GPIO_PIN_2
```

Definition at line 73 of file main.h.

6.10.2.76 **USB_DM_RESERVED_GPIO_Port**

```
#define USB_DM_RESERVED_GPIO_Port GPIOA
```

Definition at line 114 of file main.h.

6.10.2.77 **USB_DM_RESERVED_Pin**

```
#define USB_DM_RESERVED_Pin GPIO_PIN_11
```

Definition at line 113 of file main.h.

6.10.2.78 **USB_DP_RESERVED_GPIO_Port**

```
#define USB_DP_RESERVED_GPIO_Port GPIOA
```

Definition at line 116 of file main.h.

6.10.2.79 **USB_DP_RESERVED_Pin**

```
#define USB_DP_RESERVED_Pin GPIO_PIN_12
```

Definition at line 115 of file main.h.

6.10.2.80 **WATCHDOG_OUT_GPIO_Port**

```
#define WATCHDOG_OUT_GPIO_Port GPIOB
```

Definition at line 86 of file main.h.

6.10.2.81 WATCHDOG_OUT_Pin

```
#define WATCHDOG_OUT_Pin GPIO_PIN_1
```

Definition at line 85 of file main.h.

6.10.3 Function Documentation

6.10.3.1 Error_Handler()

```
void Error_Handler (
    void )
```

This function is executed in case of error occurrence.

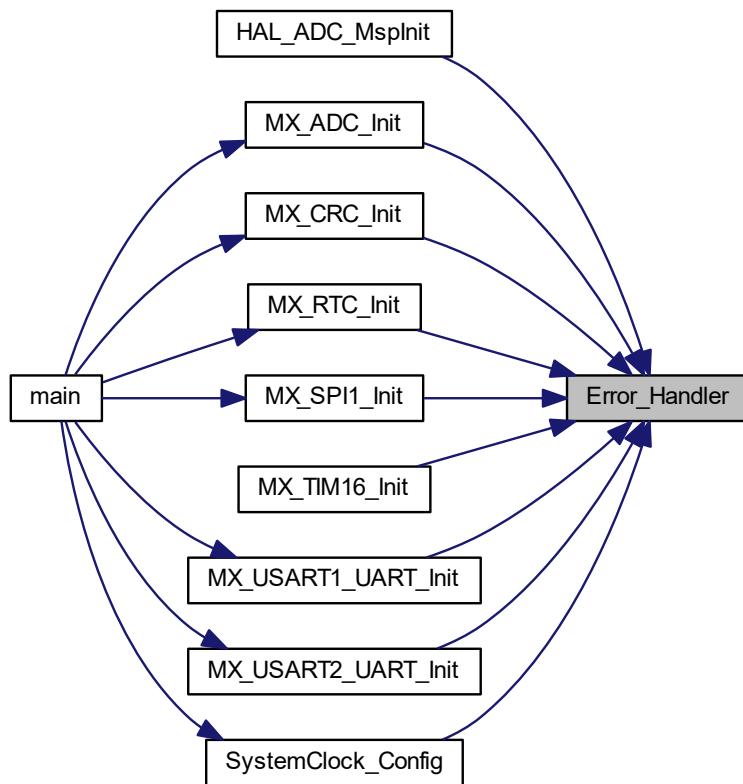
Return values

None	
------	--

Definition at line 323 of file main.cpp.

```
325 {
326     /* USER CODE BEGIN Error_Handler_Debug */
327     /* User can add his own implementation to report the HAL error return state */
328
329     /* USER CODE END Error_Handler_Debug */
```

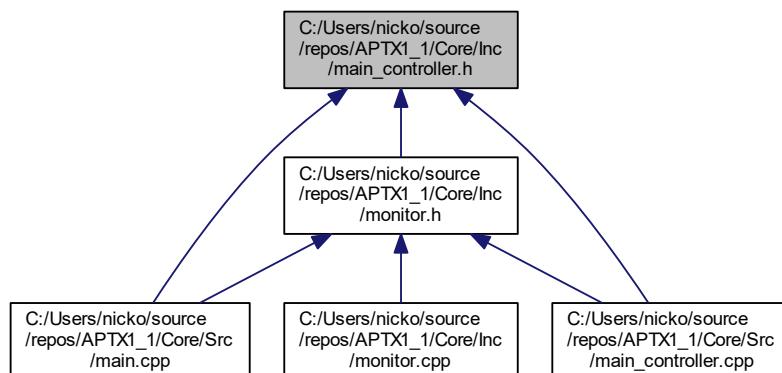
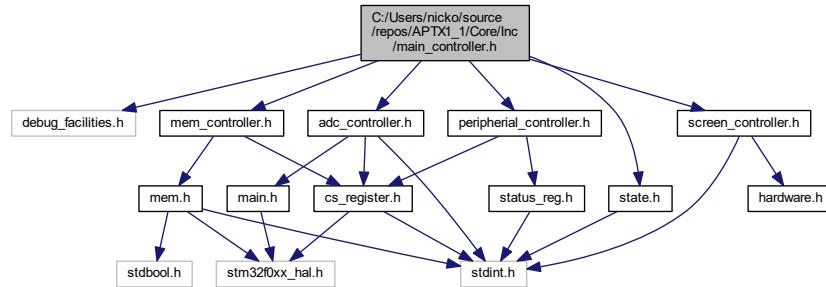
Here is the caller graph for this function:



6.11 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main_controller.h File Reference

```
#include "debug_facilities.h"
#include "peripheral_controller.h"
#include "adc_controller.h"
#include "mem_controller.h"
#include "state.h"
#include "screen_controller.h"
```

Include dependency graph for main_controller.h:



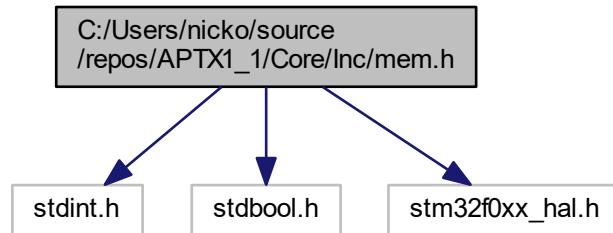
Classes

- class [MainController](#)

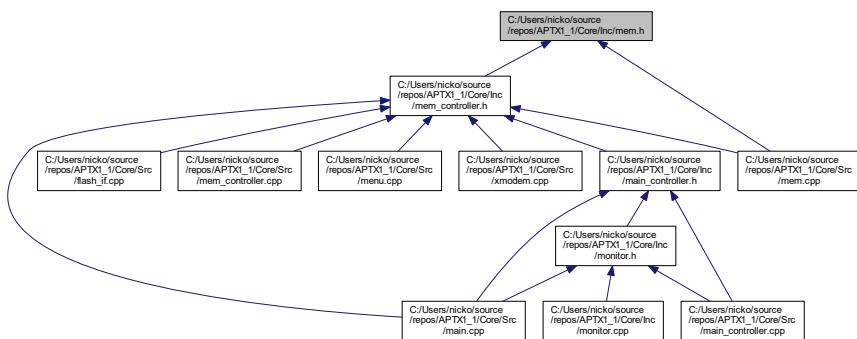
6.12 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
```

```
#include <stm32f0xx_hal.h>
Include dependency graph for mem.h:
```



This graph shows which files directly or indirectly include this file:



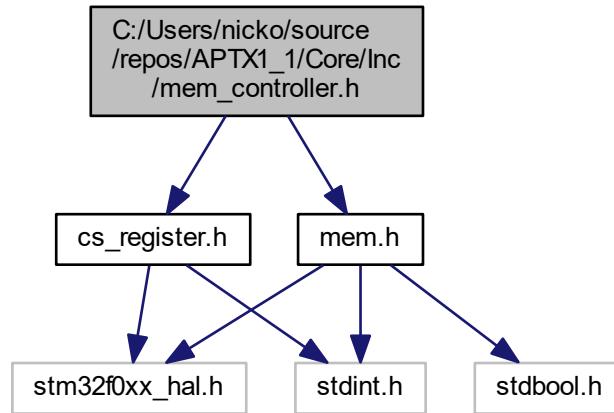
Classes

- union [byte8_t_mem_status](#)
- class [Memory](#)

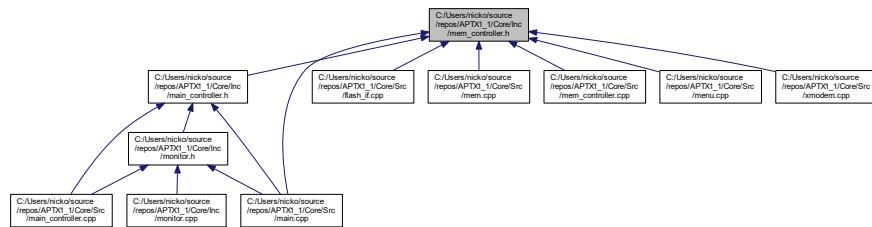
6.13 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem_controller.h File Reference

```
#include "mem.h"
#include "cs_register.h"
```

Include dependency graph for mem_controller.h:



This graph shows which files directly or indirectly include this file:



Classes

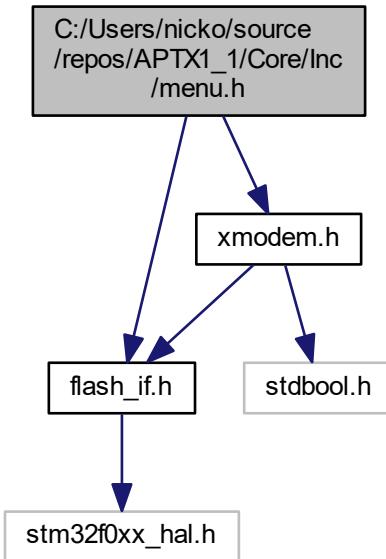
- class [Mem_ctrl](#)

6.14 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/menu.h File Reference

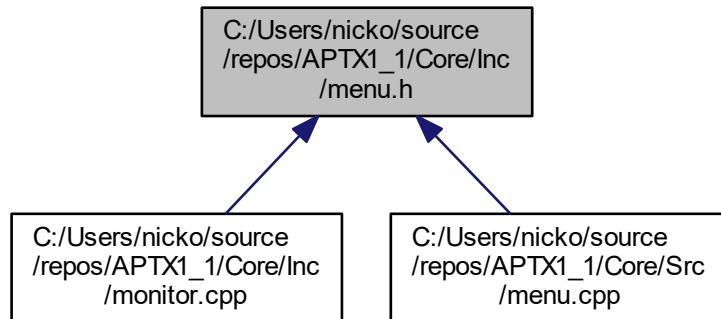
This file provides all the headers of the menu functions.

```
#include "flash_if.h"
#include "xmodem.h"
```

Include dependency graph for menu.h:



This graph shows which files directly or indirectly include this file:



Typedefs

- `typedef void(* pFunction) (void)`

Functions

- `void Main_Menu (void)`
Display the Main Menu on HyperTerminal.
- `void SerialDownload (void)`

Variables

- uint8_t aFileName [FILE_NAME_LENGTH]

6.14.1 Detailed Description

This file provides all the headers of the menu functions.

Author

MCD Application Team

Version

1.0.0

Date

8-April-2015

Attention

© COPYRIGHT(c) 2015 STMicroelectronics

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.14.2 Typedef Documentation

6.14.2.1 pFunction

```
typedef void(* pFunction) (void)
```

Definition at line 50 of file menu.h.

6.14.3 Function Documentation

6.14.3.1 SerialDownload()

```
void SerialDownload (
    void )
```

6.14.4 Variable Documentation

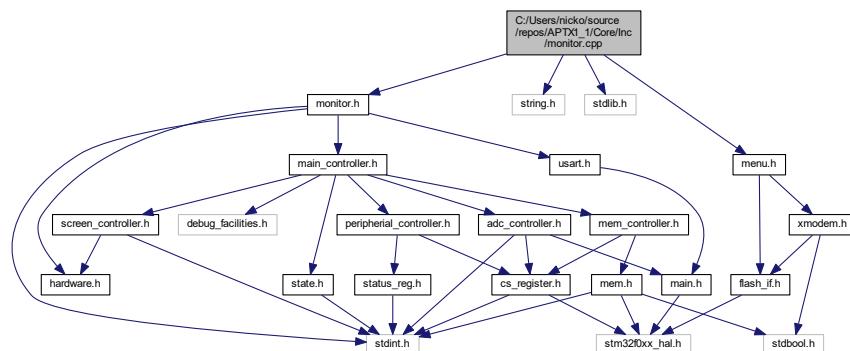
6.14.4.1 aFileName

```
uint8_t aFileName[FILE_NAME_LENGTH]
```

6.15 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.cpp File Reference

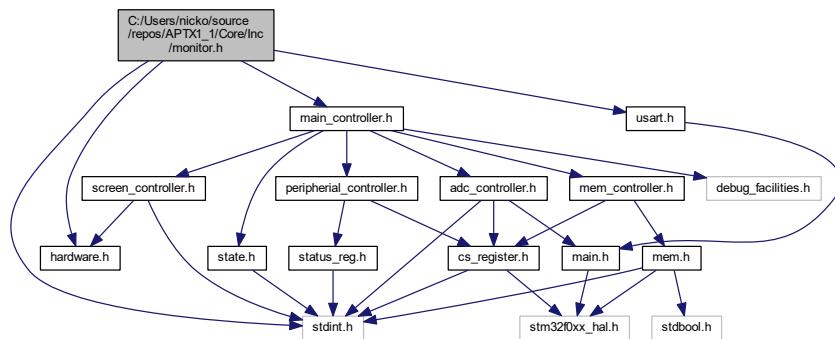
```
#include "monitor.h"
#include <string.h>
#include <stdlib.h>
#include "menu.h"

Include dependency graph for monitor.cpp:
```

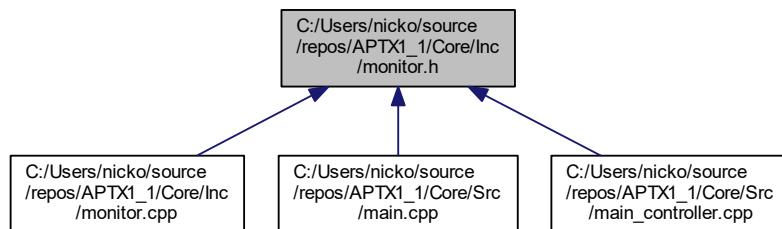


6.16 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.h File Reference

```
#include <stdint.h>
#include "hardware.h"
#include "main_controller.h"
#include "uart.h"
Include dependency graph for monitor.h:
```



This graph shows which files directly or indirectly include this file:



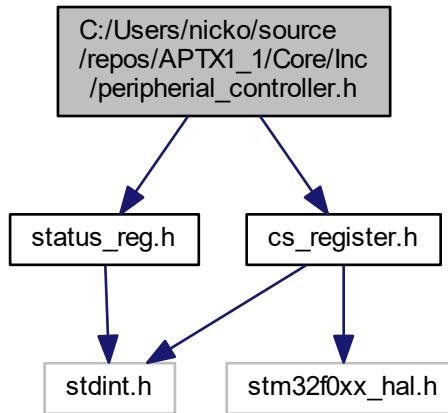
Classes

- class [Monitor](#)

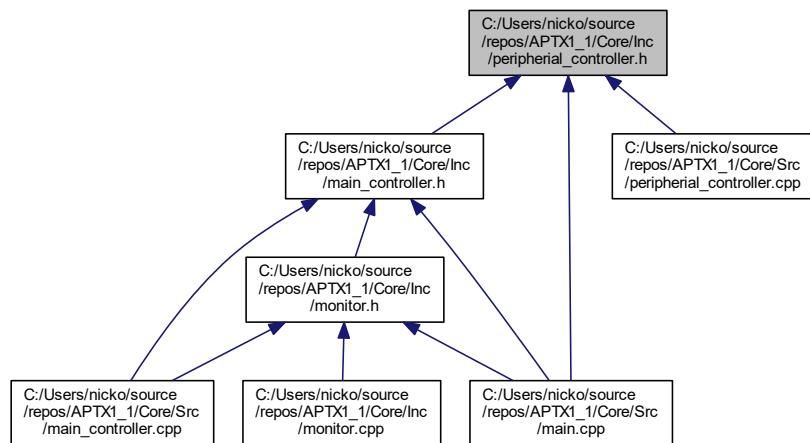
6.17 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/peripheral_controller.h File Reference

```
#include "cs_register.h"
#include "status_reg.h"
```

Include dependency graph for peripheral_controller.h:



This graph shows which files directly or indirectly include this file:



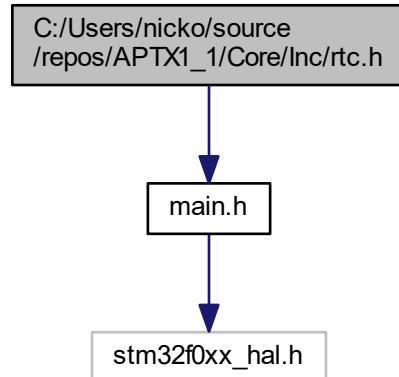
Classes

- class [PeripheralDevices](#)

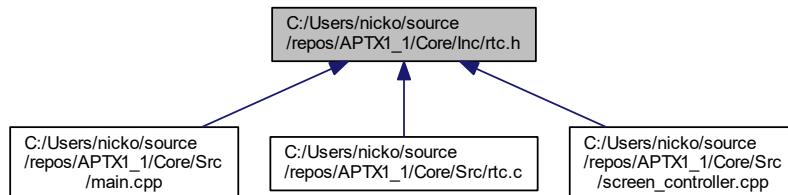
6.18 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/rtc.h File Reference

```
#include "main.h"
```

Include dependency graph for rtc.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_RTC_Init](#) (void)

Variables

- RTC_HandleTypeDef [hrtc](#)

6.18.1 Function Documentation

6.18.1.1 MX_RTC_Init()

```
void MX_RTC_Init (
    void )
```

Initialize RTC Only

Initialize RTC and set the Time and Date

Enable the TimeStamp

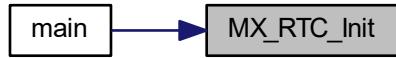
Definition at line 30 of file rtc.c.

```
31 {
32     RTC_TimeTypeDef sTime = {0};
33     RTC_DateTypeDef sDate = {0};
34
35     /** Initialize RTC Only
36     */
37     hrtc.Instance = RTC;
38     hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
39     hrtc.Init.AsynchPrediv = 127;
40     hrtc.Init.SynchPrediv = 255;
41     hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
42     hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
43     hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
44     if (HAL_RTC_Init(&hrtc) != HAL_OK)
45     {
46         Error_Handler();
47     }
48
49     /* USER CODE BEGIN Check_RTC_BKUP */
50
51     /* USER CODE END Check_RTC_BKUP */
52
53     /** Initialize RTC and set the Time and Date
54     */
55     sTime.Hours = 0x0;
56     sTime.Minutes = 0x0;
57     sTime.Seconds = 0x0;
58     sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
59     sTime.StoreOperation = RTC_STOREOPERATION_RESET;
60     if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
61     {
62         Error_Handler();
63     }
64     sDate.WeekDay = RTC_WEEKDAY_MONDAY;
65     sDate.Month = RTC_MONTH_JANUARY;
66     sDate.Date = 0x1;
67     sDate.Year = 0x0;
68
69     if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD) != HAL_OK)
70     {
71         Error_Handler();
72     }
73     /** Enable the TimeStamp
74     */
75     if (HAL_RTCEx_SetTimeStamp(&hrtc, RTC_TIMESTAMPEDGE_RISING, RTC_TIMESTAMPPIN_DEFAULT) != HAL_OK)
76     {
77         Error_Handler();
78     }
79 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.2 Variable Documentation

6.18.2.1 hrtc

RTC_HandleTypeDef hrtc

File Name : [RTC.h](#) Description : This file provides code for the configuration of the RTC instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [RTC.c](#) Description : This file provides code for the configuration of the RTC instances.

Attention

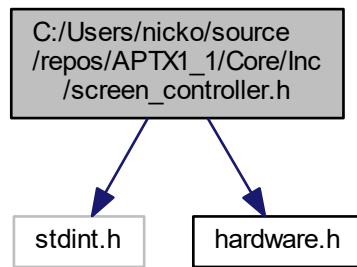
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

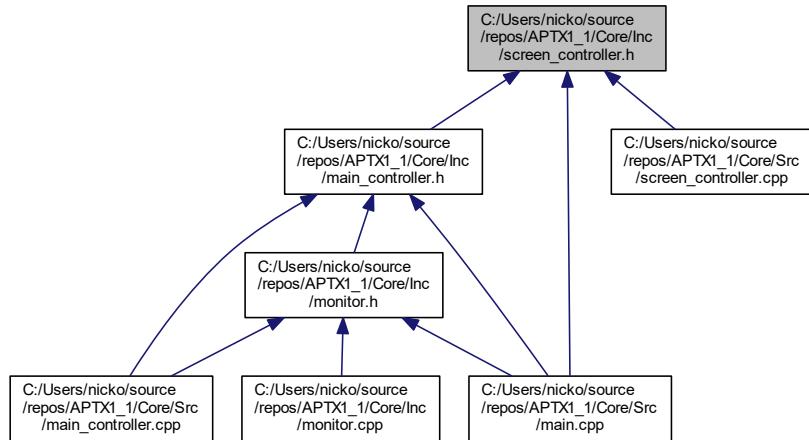
Definition at line 27 of file rtc.c.

6.19 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/screen_controller.h File Reference

```
#include <stdint.h>
#include "hardware.h"
Include dependency graph for screen_controller.h:
```



This graph shows which files directly or indirectly include this file:



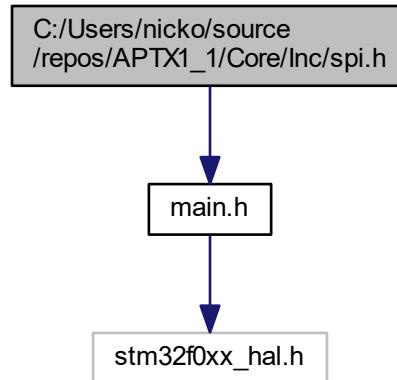
Classes

- class [Screen_ctrl](#)

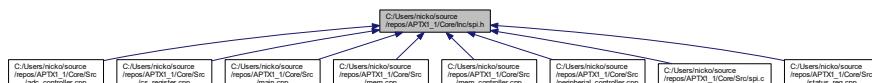
6.20 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/spi.h File Reference

```
#include "main.h"
```

Include dependency graph for spi.h:



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_SPI1_Init](#) (void)

Variables

- SPI_HandleTypeDef [hspi1](#)

6.20.1 Function Documentation

6.20.1.1 MX_SPI1_Init()

```
void MX_SPI1_Init (
    void )
```

Definition at line 30 of file spi.c.

```
31 {
32
33     hspi1.Instance = SPI1;
34     hspi1.Init.Mode = SPI_MODE_MASTER;
35     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
36     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
37     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
38     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
39     hspi1.Init.NSS = SPI NSS_SOFT;
40     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
41     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
42     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
43     hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
44     hspi1.Init.CRCPolynomial = 7;
45     hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
46     hspi1.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
47     if (HAL_SPI_Init(&hspi1) != HAL_OK)
48     {
49         Error_Handler();
50     }
51 }
52 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.20.2 Variable Documentation

6.20.2.1 hspi1

SPI_HandleTypeDef hspi1

File Name : [SPI.h](#) Description : This file provides code for the configuration of the SPI instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [SPI.c](#) Description : This file provides code for the configuration of the SPI instances.

Attention

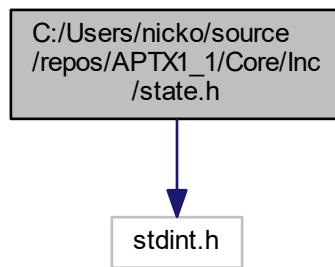
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

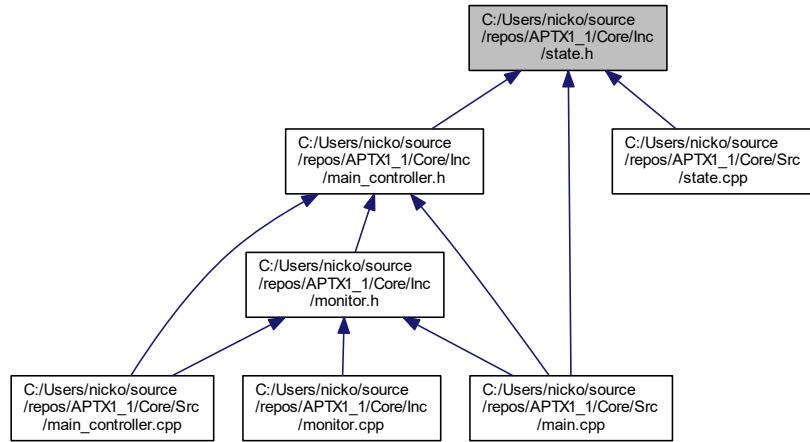
Definition at line 27 of file spi.c.

6.21 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/state.h File Reference

```
#include <stdint.h>
Include dependency graph for state.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- union `bypass_state`
- class `State`

Enumerations

- enum `statesMachine` {
 `Init`, `Idle`, `UpdateData`, `Active`,
`UartDebug`, `QA` }

6.21.1 Enumeration Type Documentation

6.21.1.1 statesMachine

```
enum statesMachine
```

Enumerator

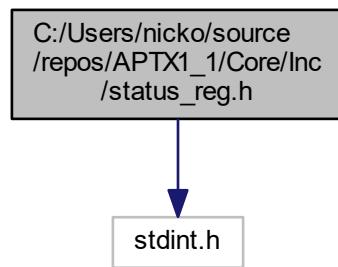
<code>Init</code>	Initialisation is still running.
<code>Idle</code>	Idling and waiting for user input.
<code>UpdateData</code>	Once in a while we send updates to screen to reduce traffic on uart1.
<code>Active</code>	Motor is running, we have to update screen and write to memories.
<code>UartDebug</code>	Special instrumentation state to when we just wait for uart4 commands. Entered by sending debug#
<code>QA</code>	

Definition at line 3 of file state.h.

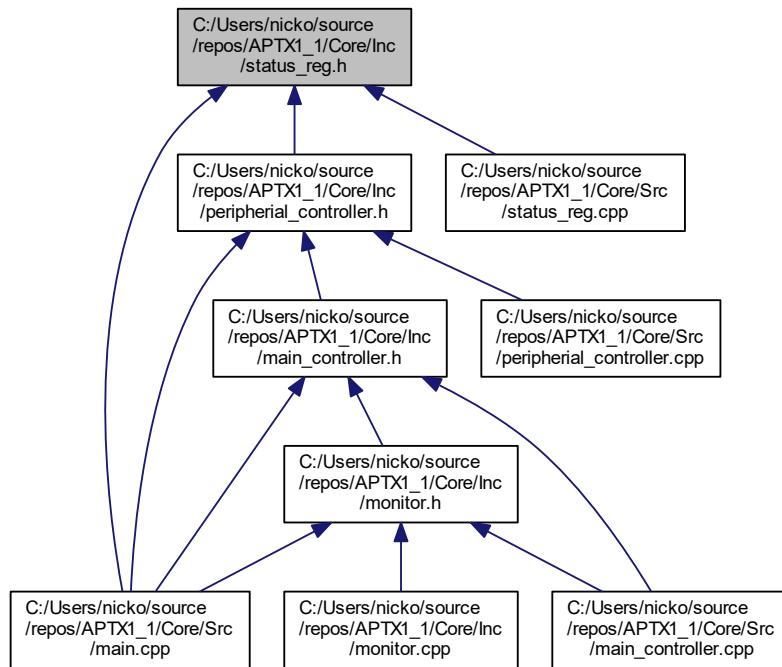
```
4 {
5     Init,
6     /**< Initialisation is still running. */
7     Idle,
8     /**< Idling and waiting for user input. */
9     UpdateData,
10    /**< Once in a while we send updates to screen to reduce traffic on uart1. */
11    Active,
12    /**< Motor is running, we have to update screen and write to memories. */
13    UartDebug,
14    /**< Special instrumentation state to when we just wait for uart4 commands. Entered by sending debug#
15    */
15    QA,
16 };
```

6.22 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/status_reg.h File Reference

```
#include <stdint.h>
Include dependency graph for status_reg.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- union `byte8_t_reg_status`
- class `STATUS_reg`

6.23 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_hal_conf.h File Reference

HAL configuration file.

```
#include "stm32f0xx_hal_rcc.h"
#include "stm32f0xx_hal_gpio.h"
#include "stm32f0xx_hal_dma.h"
#include "stm32f0xx_hal_cortex.h"
#include "stm32f0xx_hal_adc.h"
#include "stm32f0xx_hal_crc.h"
#include "stm32f0xx_hal_flash.h"
#include "stm32f0xx_hal_i2c.h"
#include "stm32f0xx_hal_pwr.h"
#include "stm32f0xx_hal_rtc.h"
#include "stm32f0xx_hal_spi.h"
#include "stm32f0xx_hal_tim.h"
```

```
#include "stm32f0xx_hal_uart.h"
Include dependency graph for stm32f0xx_hal_conf.h:
```



Macros

- `#define HAL_MODULE_ENABLED`

This is the list of modules to be used in the HAL driver.

- `#define HAL_ADC_MODULE_ENABLED`
- `#define HAL_CRC_MODULE_ENABLED`
- `#define HAL_RTC_MODULE_ENABLED`
- `#define HAL_SPI_MODULE_ENABLED`
- `#define HAL_TIM_MODULE_ENABLED`
- `#define HAL_UART_MODULE_ENABLED`
- `#define HAL_CORTEX_MODULE_ENABLED`
- `#define HAL_DMA_MODULE_ENABLED`
- `#define HAL_FLASH_MODULE_ENABLED`
- `#define HAL_GPIO_MODULE_ENABLED`
- `#define HAL_PWR_MODULE_ENABLED`
- `#define HAL_RCC_MODULE_ENABLED`
- `#define HAL_I2C_MODULE_ENABLED`
- `#define HSE_VALUE ((uint32_t)8000000)`

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

- `#define HSE_STARTUP_TIMEOUT ((uint32_t)100)`

In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.

- `#define HSI_VALUE ((uint32_t)8000000)`

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

- `#define HSI_STARTUP_TIMEOUT ((uint32_t)5000)`

In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.

- `#define HSI14_VALUE ((uint32_t)14000000)`

Internal High Speed oscillator for ADC (HSI14) value.

- `#define HSI48_VALUE ((uint32_t)48000000)`

Internal High Speed oscillator for USB (HSI48) value.

- `#define LSI_VALUE ((uint32_t)40000)`

Internal Low Speed oscillator (LSI) value.

- `#define LSE_VALUE ((uint32_t)32768)`

External Low Speed oscillator (LSI) value.

- `#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)`

- `#define VDD_VALUE ((uint32_t)3300)`

This is the HAL system configuration section.

- `#define TICK_INT_PRIORITY ((uint32_t)0)`

- `#define USERTOS 0`

- `#define PREFETCH_ENABLE 1`

- `#define INSTRUCTION_CACHE_ENABLE 0`

- `#define DATA_CACHE_ENABLE 0`

- `#define USESPI_CRC 0U`

Uncomment the line below to expand the "assert_param" macro in the HAL drivers code.

- `#define assert_param(expr) ((void)0U)`

Include module's header file.

6.23.1 Detailed Description

HAL configuration file.

Attention

© COPYRIGHT(c) 2020 STMicroelectronics

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.23.2 Macro Definition Documentation

6.23.2.1 assert_param

```
#define assert_param(  
    expr ) ((void) 0U)
```

Include module's header file.

Definition at line 318 of file stm32f0xx_hal_conf.h.

6.23.2.2 DATA_CACHE_ENABLE

```
#define DATA_CACHE_ENABLE 0
```

Definition at line 174 of file stm32f0xx_hal_conf.h.

6.23.2.3 HAL_ADC_MODULE_ENABLED

```
#define HAL_ADC_MODULE_ENABLED
```

Definition at line 51 of file stm32f0xx_hal_conf.h.

6.23.2.4 HAL_CORTEX_MODULE_ENABLED

```
#define HAL_CORTEX_MODULE_ENABLED
```

Definition at line 76 of file stm32f0xx_hal_conf.h.

6.23.2.5 HAL_CRC_MODULE_ENABLED

```
#define HAL_CRC_MODULE_ENABLED
```

Definition at line 56 of file stm32f0xx_hal_conf.h.

6.23.2.6 HAL_DMA_MODULE_ENABLED

```
#define HAL_DMA_MODULE_ENABLED
```

Definition at line 77 of file stm32f0xx_hal_conf.h.

6.23.2.7 HAL_FLASH_MODULE_ENABLED

```
#define HAL_FLASH_MODULE_ENABLED
```

Definition at line 78 of file stm32f0xx_hal_conf.h.

6.23.2.8 HAL_GPIO_MODULE_ENABLED

```
#define HAL_GPIO_MODULE_ENABLED
```

Definition at line 79 of file stm32f0xx_hal_conf.h.

6.23.2.9 HAL_I2C_MODULE_ENABLED

```
#define HAL_I2C_MODULE_ENABLED
```

Definition at line 82 of file stm32f0xx_hal_conf.h.

6.23.2.10 HAL_MODULE_ENABLED

```
#define HAL_MODULE_ENABLED
```

This is the list of modules to be used in the HAL driver.

Definition at line 50 of file stm32f0xx_hal_conf.h.

6.23.2.11 HAL_PWR_MODULE_ENABLED

```
#define HAL_PWR_MODULE_ENABLED
```

Definition at line 80 of file stm32f0xx_hal_conf.h.

6.23.2.12 HAL_RCC_MODULE_ENABLED

```
#define HAL_RCC_MODULE_ENABLED
```

Definition at line 81 of file stm32f0xx_hal_conf.h.

6.23.2.13 HAL_RTC_MODULE_ENABLED

```
#define HAL_RTC_MODULE_ENABLED
```

Definition at line 65 of file stm32f0xx_hal_conf.h.

6.23.2.14 HAL_SPI_MODULE_ENABLED

```
#define HAL_SPI_MODULE_ENABLED
```

Definition at line 66 of file stm32f0xx_hal_conf.h.

6.23.2.15 HAL_TIM_MODULE_ENABLED

```
#define HAL_TIM_MODULE_ENABLED
```

Definition at line 67 of file stm32f0xx_hal_conf.h.

6.23.2.16 HAL_UART_MODULE_ENABLED

```
#define HAL_UART_MODULE_ENABLED
```

Definition at line 68 of file stm32f0xx_hal_conf.h.

6.23.2.17 HSE_STARTUP_TIMEOUT

```
#define HSE_STARTUP_TIMEOUT ((uint32_t)100)
```

In the following line adjust the External High Speed oscillator (HSE) Startup Timeout value.

Time out for HSE start up, in ms

Definition at line 99 of file stm32f0xx_hal_conf.h.

6.23.2.18 HSE_VALUE

```
#define HSE_VALUE ((uint32_t)8000000)
```

Adjust the value of External High Speed oscillator (HSE) used in your application. This value is used by the RCC HAL module to compute the system frequency (when HSE is used as system clock source, directly or through the PLL).

Value of the External oscillator in Hz

Definition at line 91 of file stm32f0xx_hal_conf.h.

6.23.2.19 HSI14_VALUE

```
#define HSI14_VALUE ((uint32_t)14000000)
```

Internal High Speed oscillator for ADC (HSI14) value.

Value of the Internal High Speed oscillator for ADC in Hz. The real value may vary depending on the variations in voltage and temperature.

Definition at line 123 of file stm32f0xx_hal_conf.h.

6.23.2.20 HSI48_VALUE

```
#define HSI48_VALUE ((uint32_t)48000000)
```

Internal High Speed oscillator for USB (HSI48) value.

Value of the Internal High Speed oscillator for USB in Hz. The real value may vary depending on the variations in voltage and temperature.

Definition at line 134 of file stm32f0xx_hal_conf.h.

6.23.2.21 HSI_STARTUP_TIMEOUT

```
#define HSI_STARTUP_TIMEOUT ((uint32_t)5000)
```

In the following line adjust the Internal High Speed oscillator (HSI) Startup Timeout value.

Time out for HSI start up

Definition at line 116 of file stm32f0xx_hal_conf.h.

6.23.2.22 HSI_VALUE

```
#define HSI_VALUE ((uint32_t)8000000)
```

Internal High Speed oscillator (HSI) value. This value is used by the RCC HAL module to compute the system frequency (when HSI is used as system clock source, directly or through the PLL).

Value of the Internal oscillator in Hz

Definition at line 108 of file stm32f0xx_hal_conf.h.

6.23.2.23 INSTRUCTION_CACHE_ENABLE

```
#define INSTRUCTION_CACHE_ENABLE 0
```

Definition at line 173 of file stm32f0xx_hal_conf.h.

6.23.2.24 LSE_STARTUP_TIMEOUT

```
#define LSE_STARTUP_TIMEOUT ((uint32_t)5000)
```

Time out for LSE start up, in ms

Definition at line 157 of file stm32f0xx_hal_conf.h.

6.23.2.25 LSE_VALUE

```
#define LSE_VALUE ((uint32_t)32768)
```

External Low Speed oscillator (LSI) value.

< Value of the Internal Low Speed oscillator in Hz The real value may vary depending on the variations in voltage and temperature.

Value of the External Low Speed oscillator in Hz

Definition at line 153 of file stm32f0xx_hal_conf.h.

6.23.2.26 LSI_VALUE

```
#define LSI_VALUE ((uint32_t)40000)
```

Internal Low Speed oscillator (LSI) value.

Definition at line 145 of file stm32f0xx_hal_conf.h.

6.23.2.27 PREFETCH_ENABLE

```
#define PREFETCH_ENABLE 1
```

Definition at line 172 of file stm32f0xx_hal_conf.h.

6.23.2.28 TICK_INT_PRIORITY

```
#define TICK_INT_PRIORITY ((uint32_t)0)
```

tick interrupt priority (lowest by default)

Definition at line 168 of file stm32f0xx_hal_conf.h.

6.23.2.29 USE_RTOS

```
#define USE_RTOS 0
```

Definition at line 171 of file stm32f0xx_hal_conf.h.

6.23.2.30 USE_SPI_CRC

```
#define USE_SPI_CRC 0U
```

Uncomment the line below to expand the "assert_param" macro in the HAL drivers code.

Definition at line 189 of file stm32f0xx_hal_conf.h.

6.23.2.31 VDD_VALUE

```
#define VDD_VALUE ((uint32_t)3300)
```

This is the HAL system configuration section.

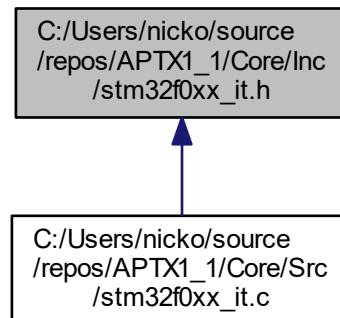
Value of VDD in mv

Definition at line 167 of file stm32f0xx_hal_conf.h.

6.24 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_it.h File Reference

This file contains the headers of the interrupt handlers.

This graph shows which files directly or indirectly include this file:



Functions

- void [NMI_Handler](#) (void)
This function handles Non maskable interrupt.
- void [HardFault_Handler](#) (void)
This function handles Hard fault interrupt.
- void [SVC_Handler](#) (void)
This function handles System service call via SWI instruction.
- void [PendSV_Handler](#) (void)
This function handles Pendable request for system service.
- void [SysTick_Handler](#) (void)
This function handles System tick timer.
- void [DMA1_Channel1_IRQHandler](#) (void)
This function handles DMA1 channel 1 global interrupt.
- void [TIM1_BRK_UP_TRG_COM_IRQHandler](#) (void)
This function handles TIM1 break, update, trigger and commutation interrupts.
- void [TIM16_IRQHandler](#) (void)
This function handles TIM16 global interrupt.
- void [USART1_IRQHandler](#) (void)
This function handles USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25.

6.24.1 Detailed Description

This file contains the headers of the interrupt handlers.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

6.24.2 Function Documentation

6.24.2.1 DMA1_Channel1_IRQHandler()

```
void DMA1_Channel1_IRQHandler (
    void )
```

This function handles DMA1 channel 1 global interrupt.

Definition at line 167 of file stm32f0xx_it.c.

```
168 {
169     /* USER CODE BEGIN DMA1_Channel1_IRQHandler_0 */
170
171     /* USER CODE END DMA1_Channel1_IRQHandler_0 */
172     HAL_DMA_IRQHandler(&hdma_adc);
173     /* USER CODE BEGIN DMA1_Channel1_IRQHandler_1 */
174
175     /* USER CODE END DMA1_Channel1_IRQHandler_1 */
176 }
```

6.24.2.2 HardFault_Handler()

```
void HardFault_Handler (
    void )
```

This function handles Hard fault interrupt.

Definition at line 105 of file stm32f0xx_it.c.

```
106 {
107     /* USER CODE BEGIN HardFault_IRQHandler_0 */
108
109     /* USER CODE END HardFault_IRQHandler_0 */
110     while (1)
111     {
112         /* USER CODE BEGIN W1_HardFault_IRQHandler_0 */
113         /* USER CODE END W1_HardFault_IRQHandler_0 */
114     }
115 }
```

6.24.2.3 NMI_Handler()

```
void NMI_Handler (
    void )
```

This function handles Non maskable interrupt.

Definition at line 92 of file stm32f0xx_it.c.

```
93 {
94     /* USER CODE BEGIN NonMaskableInt_IRQn 0 */
95
96     /* USER CODE END NonMaskableInt_IRQn 0 */
97     /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
98
99     /* USER CODE END NonMaskableInt_IRQn 1 */
100 }
```

6.24.2.4 PendSV_Handler()

```
void PendSV_Handler (
    void )
```

This function handles Pendable request for system service.

Definition at line 133 of file stm32f0xx_it.c.

```
134 {
135     /* USER CODE BEGIN PendSV_IRQn 0 */
136
137     /* USER CODE END PendSV_IRQn 0 */
138     /* USER CODE BEGIN PendSV_IRQn 1 */
139
140     /* USER CODE END PendSV_IRQn 1 */
141 }
```

6.24.2.5 SVC_Handler()

```
void SVC_Handler (
    void )
```

This function handles System service call via SWI instruction.

Definition at line 120 of file stm32f0xx_it.c.

```
121 {
122     /* USER CODE BEGIN SVC_IRQn 0 */
123
124     /* USER CODE END SVC_IRQn 0 */
125     /* USER CODE BEGIN SVC_IRQn 1 */
126
127     /* USER CODE END SVC_IRQn 1 */
128 }
```

6.24.2.6 SysTick_Handler()

```
void SysTick_Handler (
    void )
```

This function handles System tick timer.

Definition at line 146 of file stm32f0xx_it.c.

```
147 {
148     /* USER CODE BEGIN SysTick_IRQn 0 */
149
150     /* USER CODE END SysTick_IRQn 0 */
151
152     /* USER CODE BEGIN SysTick_IRQn 1 */
153
154     /* USER CODE END SysTick_IRQn 1 */
155 }
```

6.24.2.7 TIM16_IRQHandler()

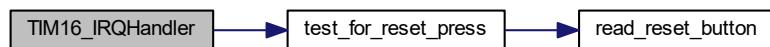
```
void TIM16_IRQHandler (
    void )
```

This function handles TIM16 global interrupt.

Definition at line 195 of file stm32f0xx_it.c.

```
196 {
197     /* USER CODE BEGIN TIM16_IRQn 0 */
198
199     if (test_for_reset_press)
200     {
201         reset_pressed = true;
202     }
203
204     /* USER CODE END TIM16_IRQn 0 */
205     HAL_TIM_IRQHandler(&htim16);
206     /* USER CODE BEGIN TIM16_IRQn 1 */
207
208     /* USER CODE END TIM16_IRQn 1 */
209 }
```

Here is the call graph for this function:



6.24.2.8 TIM1_BRK_UP_TRG_COM_IRQHandler()

```
void TIM1_BRK_UP_TRG_COM_IRQHandler (
    void )
```

This function handles TIM1 break, update, trigger and commutation interrupts.

Definition at line 181 of file stm32f0xx_it.c.

```
182 {
183     /* USER CODE BEGIN TIM1_BRK_UP_TRG_COM_IRQHandler_0 */
184
185     /* USER CODE END TIM1_BRK_UP_TRG_COM_IRQHandler_0 */
186     HAL_TIM_IRQHandler(&htim1);
187     /* USER CODE BEGIN TIM1_BRK_UP_TRG_COM_IRQHandler_1 */
188
189     /* USER CODE END TIM1_BRK_UP_TRG_COM_IRQHandler_1 */
190 }
```

6.24.2.9 USART1_IRQHandler()

```
void USART1_IRQHandler (
    void )
```

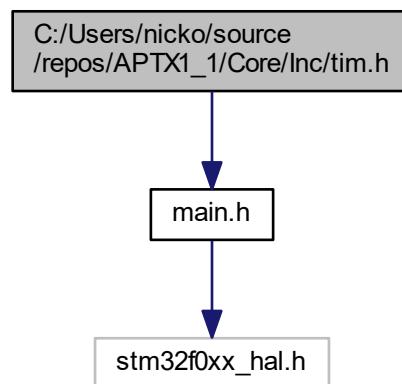
This function handles USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25.

Definition at line 214 of file stm32f0xx_it.c.

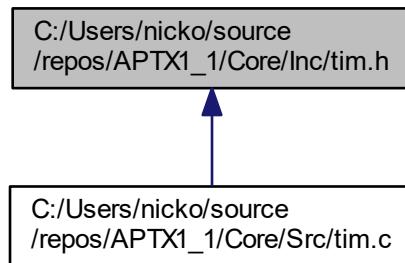
```
215 {
216     /* USER CODE BEGIN USART1_IRQHandler_0 */
217
218     /* USER CODE END USART1_IRQHandler_0 */
219     HAL_UART_IRQHandler(&huart1);
220     /* USER CODE BEGIN USART1_IRQHandler_1 */
221
222     /* USER CODE END USART1_IRQHandler_1 */
223 }
```

6.25 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/tim.h File Reference

```
#include "main.h"
Include dependency graph for tim.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void [MX_TIM16_Init](#) (void)

Variables

- TIM_HandleTypeDef [htim16](#)

6.25.1 Function Documentation

6.25.1.1 MX_TIM16_Init()

```
void MX_TIM16_Init (
    void )
```

Definition at line 30 of file tim.c.

```
31 {
32     TIM_OC_InitTypeDef sConfigOC = {0};
33     TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};
34
35     htim16.Instance = TIM16;
36     htim16.Init.Prescaler = 48000;
37     htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
38     htim16.Init.Period = 10;
39     htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
40     htim16.Init.RepetitionCounter = 5;
41     htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
42     if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
43     {
44         Error_Handler();
45     }
46     if (HAL_TIM_OC_Init(&htim16) != HAL_OK)
47     {
48         Error_Handler();
49     }
50     sConfigOC.OCMode = TIM_OCMODE_TIMING;
51     sConfigOC.Pulse = 0;
52     sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
53     sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
```

```

54     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
55     sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
56     sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
57     if (HAL_TIM_OC_ConfigChannel(&htim16, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
58     {
59         Error_Handler();
60     }
61     sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
62     sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
63     sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
64     sBreakDeadTimeConfig.DeadTime = 0;
65     sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
66     sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
67     sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
68     if (HAL_TIMEx_ConfigBreakDeadTime(&htim16, &sBreakDeadTimeConfig) != HAL_OK)
69     {
70         Error_Handler();
71     }
72 }
73 }
```

Here is the call graph for this function:



6.25.2 Variable Documentation

6.25.2.1 htim16

TIM_HandleTypeDef htim16

File Name : [TIM.h](#) Description : This file provides code for the configuration of the TIM instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [TIM.c](#) Description : This file provides code for the configuration of the TIM instances.

Attention

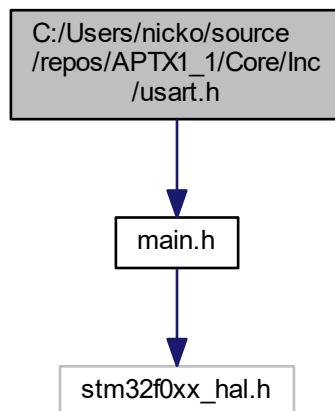
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

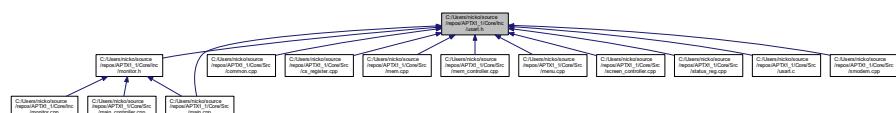
Definition at line 27 of file tim.c.

6.26 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/usart.h File Reference

```
#include "main.h"
Include dependency graph for usart.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void **MX_USART1_UART_Init** (void)
 - void **MX_USART2_UART_Init** (void)

Variables

- UART_HandleTypeDef [huart1](#)
- UART_HandleTypeDef [huart2](#)

6.26.1 Function Documentation

6.26.1.1 MX_USART1_UART_Init()

```
void MX_USART1_UART_Init (
    void )
```

Definition at line 32 of file `uart.c`.

```
33 {
34
35     huart1.Instance = USART1;
36     huart1.Init.BaudRate = 115200;
37     huart1.Init.WordLength = UART_WORDLENGTH_8B;
38     huart1.Init.StopBits = UART_STOPBITS_1;
39     huart1.Init.Parity = UART_PARITY_NONE;
40     huart1.Init.Mode = UART_MODE_TX_RX;
41     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
42     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
43     huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
44     huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
45     if (HAL_UART_Init(&huart1) != HAL_OK)
46     {
47         Error_Handler();
48     }
49
50 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.1.2 MX_USART2_UART_Init()

```
void MX_USART2_UART_Init (
    void )
```

Definition at line 53 of file usart.c.

```
54 {
55
56     huart2.Instance = USART2;
57     huart2.Init.BaudRate = 38400;
58     huart2.Init.WordLength = UART_WORDLENGTH_8B;
59     huart2.Init.StopBits = UART_STOPBITS_1;
60     huart2.Init.Parity = UART_PARITY_NONE;
61     huart2.Init.Mode = UART_MODE_TX_RX;
62     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
63     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
64     huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
65     huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
66     if (HAL_UART_Init(&huart2) != HAL_OK)
67     {
68         Error_Handler();
69     }
70
71 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.2 Variable Documentation

6.26.2.1 huart1

```
UART_HandleTypeDef huart1
```

File Name : [USART.h](#) Description : This file provides code for the configuration of the USART instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

File Name : [USART.c](#) Description : This file provides code for the configuration of the USART instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file `usart.c`.

6.26.2.2 huart2

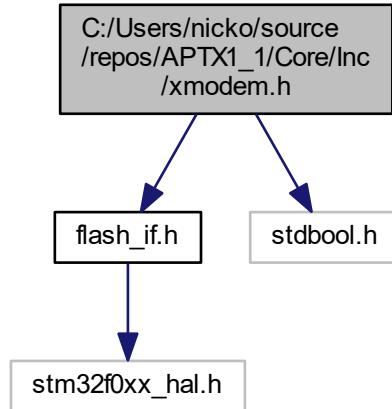
```
UART_HandleTypeDef huart2
```

Definition at line 28 of file `usart.c`.

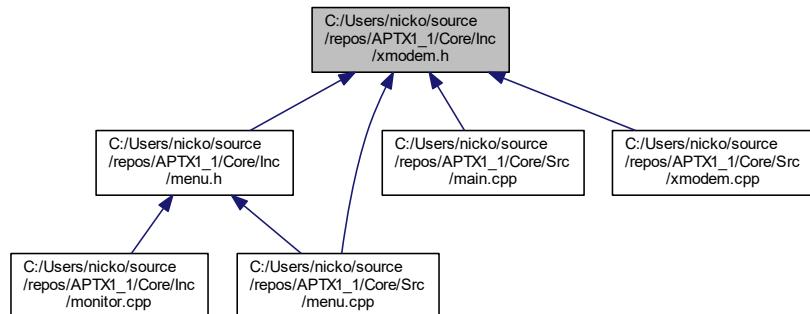
6.27 C:/Users/nicko/source/repos/APTX1_1/Core/Inc/xmodem.h File Reference

This module is the implementation of the Xmodem protocol.

```
#include "flash_if.h"
#include "stdbool.h"
Include dependency graph for xmodem.h:
```



This graph shows which files directly or indirectly include this file:



Macros

- #define X_MAX_ERRORS ((uint8_t)3u)
- #define X_PACKET_128_SIZE ((uint16_t)128u)
- #define X_PACKET_1024_SIZE ((uint16_t)1024u)
- #define X_PACKET_CRC_SIZE ((uint16_t)2u)
- #define X_PACKET_NUMBER_INDEX ((uint16_t)0u)
- #define X_PACKET_NUMBER_COMPLEMENT_INDEX ((uint16_t)1u)
- #define X_PACKET_DATA_INDEX ((uint16_t)2u)
- #define X_SOH ((uint8_t)0x01u)
- #define X_STX ((uint8_t)0x02u)
- #define X_EOT ((uint8_t)0x04u)

- #define **X_ACK** ((uint8_t)0x06u)
- #define **X_NAK** ((uint8_t)0x15u)
- #define **X_CAN** ((uint8_t)0x18u)
- #define **X_C** ((uint8_t)0x43u)
- #define **PACKET_HEADER_SIZE** ((uint32_t)3)
- #define **PACKET_DATA_INDEX** ((uint32_t)3)
- #define **PACKET_START_INDEX** ((uint32_t)0)
- #define **PACKET_NUMBER_INDEX** ((uint32_t)1)
- #define **PACKET_CNUMBER_INDEX** ((uint32_t)2)
- #define **PACKET_TRAILER_SIZE** ((uint32_t)2)
- #define **PACKET_OVERHEAD_SIZE** (**PACKET_HEADER_SIZE** + **PACKET_TRAILER_SIZE** - 1)
- #define **PACKET_SIZE** ((uint32_t)128)
- #define **PACKET_1K_SIZE** ((uint32_t)1024)
- #define **FILE_NAME_LENGTH** ((uint32_t)64)
- #define **FILE_SIZE_LENGTH** ((uint32_t)16)
- #define **SOH** ((uint8_t)0x01) /* start of 128-byte data packet */
- #define **STX** ((uint8_t)0x02) /* start of 1024-byte data packet */
- #define **EOT** ((uint8_t)0x04) /* end of transmission */
- #define **ACK** ((uint8_t)0x06) /* acknowledge */
- #define **NAK** ((uint8_t)0x15) /* negative acknowledge */
- #define **CA** ((uint8_t)0x18) /* two of these in succession aborts transfer */
- #define **CRC16** ((uint8_t)0x43) /* 'C' == 0x43, request 16-bit CRC */
- #define **NEGATIVE_BYTE** ((uint8_t)0xFF)
- #define **ABORT1** ((uint8_t)0x41) /* 'A' == 0x41, abort by user */
- #define **ABORT2** ((uint8_t)0x61) /* 'a' == 0x61, abort by user */
- #define **NAK_TIMEOUT** ((uint32_t)0x100000)
- #define **DOWNLOAD_TIMEOUT** ((uint32_t)1000) /* One second retry delay */
- #define **MAX_ERRORS** ((uint32_t)50)

Enumerations

- enum **xmodem_status** {
 X_OK = 0x00u, **X_ERROR_CRC** = 0x01u, **X_ERROR_NUMBER** = 0x02u, **X_ERROR_UART** = 0x04u,
 X_ERROR_FLASH = 0x06u, **X_ERROR** = 0xFFu
 }

Functions

- void **xmodem_receive** (void)

This function is the base of the Xmodem protocol. When we receive a header from UART, it decides what action it shall take.
- uint16_t **xmodem_calc_crc** (uint8_t *data, uint16_t length)

Calculates the CRC-16 for the input package.

6.27.1 Detailed Description

This module is the implementation of the Xmodem protocol.

Author

Ferenc Nemeth

Date

21 Dec 2018

Copyright (c) 2018 Ferenc Nemeth - <https://github.com/ferenc-nemeth>

6.27.2 Macro Definition Documentation

6.27.2.1 ABORT1

```
#define ABORT1 ((uint8_t)0x41) /* 'A' == 0x41, abort by user */
```

Definition at line 94 of file xmodem.h.

6.27.2.2 ABORT2

```
#define ABORT2 ((uint8_t)0x61) /* 'a' == 0x61, abort by user */
```

Definition at line 95 of file xmodem.h.

6.27.2.3 ACK

```
#define ACK ((uint8_t)0x06) /* acknowledge */
```

Definition at line 88 of file xmodem.h.

6.27.2.4 CA

```
#define CA ((uint8_t)0x18) /* two of these in succession aborts transfer */
```

Definition at line 90 of file xmodem.h.

6.27.2.5 CRC16

```
#define CRC16 ((uint8_t)0x43) /* 'C' == 0x43, request 16-bit CRC */
```

Definition at line 91 of file xmodem.h.

6.27.2.6 DOWNLOAD_TIMEOUT

```
#define DOWNLOAD_TIMEOUT ((uint32_t)1000) /* One second retry delay */
```

Definition at line 98 of file xmodem.h.

6.27.2.7 EOT

```
#define EOT ((uint8_t)0x04) /* end of transmission */
```

Definition at line 87 of file xmodem.h.

6.27.2.8 FILE_NAME_LENGTH

```
#define FILE_NAME_LENGTH ((uint32_t)64)
```

Definition at line 82 of file xmodem.h.

6.27.2.9 FILE_SIZE_LENGTH

```
#define FILE_SIZE_LENGTH ((uint32_t)16)
```

Definition at line 83 of file xmodem.h.

6.27.2.10 MAX_ERRORS

```
#define MAX_ERRORS ((uint32_t)50)
```

Definition at line 99 of file xmodem.h.

6.27.2.11 NAK

```
#define NAK ((uint8_t)0x15) /* negative acknowledge */
```

Definition at line 89 of file xmodem.h.

6.27.2.12 NAK_TIMEOUT

```
#define NAK_TIMEOUT ((uint32_t)0x100000)
```

Definition at line 97 of file xmodem.h.

6.27.2.13 NEGATIVE_BYTE

```
#define NEGATIVE_BYTE ((uint8_t)0xFF)
```

Definition at line 92 of file xmodem.h.

6.27.2.14 PACKET_1K_SIZE

```
#define PACKET_1K_SIZE ((uint32_t)1024)
```

Definition at line 73 of file xmodem.h.

6.27.2.15 PACKET_CNUMBER_INDEX

```
#define PACKET_CNUMBER_INDEX ((uint32_t)2)
```

Definition at line 69 of file xmodem.h.

6.27.2.16 PACKET_DATA_INDEX

```
#define PACKET_DATA_INDEX ((uint32_t)3)
```

Definition at line 66 of file xmodem.h.

6.27.2.17 PACKET_HEADER_SIZE

```
#define PACKET_HEADER_SIZE ((uint32_t)3)
```

Definition at line 65 of file xmodem.h.

6.27.2.18 PACKET_NUMBER_INDEX

```
#define PACKET_NUMBER_INDEX ((uint32_t)1)
```

Definition at line 68 of file xmodem.h.

6.27.2.19 PACKET_OVERHEAD_SIZE

```
#define PACKET_OVERHEAD_SIZE (PACKET_HEADER_SIZE + PACKET_TRAILER_SIZE - 1)
```

Definition at line 71 of file xmodem.h.

6.27.2.20 PACKET_SIZE

```
#define PACKET_SIZE ((uint32_t)128)
```

Definition at line 72 of file xmodem.h.

6.27.2.21 PACKET_START_INDEX

```
#define PACKET_START_INDEX ((uint32_t)0)
```

Definition at line 67 of file xmodem.h.

6.27.2.22 PACKET_TRAILER_SIZE

```
#define PACKET_TRAILER_SIZE ((uint32_t)2)
```

Definition at line 70 of file xmodem.h.

6.27.2.23 SOH

```
#define SOH ((uint8_t)0x01) /* start of 128-byte data packet */
```

Definition at line 85 of file xmodem.h.

6.27.2.24 STX

```
#define STX ((uint8_t)0x02) /* start of 1024-byte data packet */
```

Definition at line 86 of file xmodem.h.

6.27.2.25 X_ACK

```
#define X_ACK ((uint8_t)0x06u)
```

Acknowledge.

Definition at line 49 of file xmodem.h.

6.27.2.26 X_C

```
#define X_C ((uint8_t)0x43u)
```

ASCII "C", to notify the host, we want to use CRC16.

Definition at line 52 of file xmodem.h.

6.27.2.27 X_CAN

```
#define X_CAN ((uint8_t)0x18u)
```

Cancel.

Definition at line 51 of file xmodem.h.

6.27.2.28 X_EOT

```
#define X_EOT ((uint8_t)0x04u)
```

End Of Transmission.

Definition at line 48 of file xmodem.h.

6.27.2.29 X_MAX_ERRORS

```
#define X_MAX_ERRORS ((uint8_t)3u)
```

Definition at line 33 of file xmodem.h.

6.27.2.30 X_NAK

```
#define X_NAK ((uint8_t)0x15u)
```

Not Acknowledge.

Definition at line 50 of file xmodem.h.

6.27.2.31 X_PACKET_1024_SIZE

```
#define X_PACKET_1024_SIZE ((uint16_t)1024u)
```

Definition at line 37 of file xmodem.h.

6.27.2.32 X_PACKET_128_SIZE

```
#define X_PACKET_128_SIZE ((uint16_t)128u)
```

Definition at line 36 of file xmodem.h.

6.27.2.33 X_PACKET_CRC_SIZE

```
#define X_PACKET_CRC_SIZE ((uint16_t)2u)
```

Definition at line 38 of file xmodem.h.

6.27.2.34 X_PACKET_DATA_INDEX

```
#define X_PACKET_DATA_INDEX ((uint16_t)2u)
```

Definition at line 43 of file xmodem.h.

6.27.2.35 X_PACKET_NUMBER_COMPLEMENT_INDEX

```
#define X_PACKET_NUMBER_COMPLEMENT_INDEX ((uint16_t)1u)
```

Definition at line 42 of file xmodem.h.

6.27.2.36 X_PACKET_NUMBER_INDEX

```
#define X_PACKET_NUMBER_INDEX ((uint16_t)0u)
```

Definition at line 41 of file xmodem.h.

6.27.2.37 X_SOH

```
#define X_SOH ((uint8_t)0x01u)
```

Start Of Header (128 bytes).

Definition at line 46 of file xmodem.h.

6.27.2.38 X_STX

```
#define X_STX ((uint8_t)0x02u)
```

Start Of Header (1024 bytes).

Definition at line 47 of file xmodem.h.

6.27.3 Enumeration Type Documentation

6.27.3.1 xmodem_status

```
enum xmodem_status
```

Enumerator

X_OK	The action was successful.
X_ERROR_CRC	CRC calculation error.
X_ERROR_NUMBER	Packet number mismatch error.
X_ERROR_UART	UART communication error.
X_ERROR_FLASH	Flash related error.
X_ERROR	Generic error.

Generated by Doxygen

Definition at line 55 of file xmodem.h.

```

55      {
56      X_OK      = 0x00u, /*<< The action was successful. */
57      X_ERROR_CRC = 0x01u, /*<< CRC calculation error. */
58      X_ERROR_NUMBER = 0x02u, /*<< Packet number mismatch error. */
59      X_ERROR_UART = 0x04u, /*<< UART communication error. */
60      X_ERROR_FLASH = 0x06u, /*<< Flash related error. */
61      X_ERROR      = 0xFFu /*<< Generic error. */
62 } xmodem_status;

```

6.27.4 Function Documentation

6.27.4.1 xmodem_calc_crc()

```

uint16_t xmodem_calc_crc (
    uint8_t * data,
    uint16_t length )

```

Calculates the CRC-16 for the input package.

Parameters

* <i>data</i>	Array of the data which we want to calculate.
<i>length</i>	Size of the data, either 128 or 1024 bytes.

Returns

status: The calculated CRC.

Definition at line 155 of file xmodem.cpp.

```

156 {
157     uint16_t crc = 0u;
158     while (length)
159     {
160         length--;
161         crc = crc ^ ((uint16_t)*data++ << 8u);
162         for (uint8_t i = 0u; i < 8u; i++)
163         {
164             if (crc & 0x8000u)
165             {
166                 crc = (crc << 1u) ^ 0x1021u;
167             }
168             else
169             {
170                 crc = crc << 1u;
171             }
172         }
173     }
174     return crc;
175 }

```

6.27.4.2 xmodem_receive()

```

void xmodem_receive (
    void )

```

This function is the base of the Xmodem protocol. When we receive a header from UART, it decides what action it shall take.

Parameters

<code>void</code>	<input type="button" value=""/>
-------------------	---------------------------------

Returns`void`**Definition at line 34 of file xmodem.cpp.**

```

35 {
36     volatile xmodem_status status = X_OK;
37     uint8_t error_number = 0u;
38
39     x_first_packet_received = false;
40     xmodem_packet_number = lu;
41     xmodem_actual_flash_address = APPLICATION_ADDRESS;
42
43     /* Loop until there isn't any error (or until we jump to the user application). */
44     while (X_OK == status)
45     {
46         uint8_t header = 0x00u;
47
48         /* Get the header from UART. */
49         HAL_StatusTypeDef comm_status = HAL_UART_Receive(SERIAL_PC, &header, lu, 100);
50
51         /* Spam the host (until we receive something) with ACSII "C", to notify it, we want to use
52          CRC-16. */
53         if ((HAL_OK != comm_status) && (false == x_first_packet_received))
54         {
55             (void)Serial_PutByte(X_C);
56
57             /* Uart timeout or any other errors. */
58             else if ((HAL_OK != comm_status) && (true == x_first_packet_received))
59             {
60                 status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
61             }
62             else
63             {
64                 /* Do nothing. */
65             }
66
67             xmodem_status packet_status = X_ERROR;
68             packet_status = xmodem_handle_packet(header);
69             /* The header can be: SOH, STX, EOT and CAN. */
70
71             switch (header)
72             {
73                 /* 128 or 1024 bytes of data. */
74                 case X_SOH:
75                     /* If the handling was successful, then send an ACK. */
76                     if (X_OK == packet_status)
77                     {
78                         (void)Serial_PutByte(X_ACK);
79
80                     /* If the error was flash related, then immediately set the error counter to max (graceful
81                      abort). */
82                     else if (X_ERROR_FLASH == packet_status)
83                     {
84                         error_number = X_MAX_ERRORS;
85                         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
86                     }
87                     /* Error while processing the packet, either send a NAK or do graceful abort. */
88                     else
89                     {
90                         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
91                     }
92                     break;
93                 case X_STX:
94                     /* If the handling was successful, then send an ACK. */
95                     if (X_OK == packet_status)
96                     {
97                         (void)Serial_PutByte(X_ACK);
98
99                     /* If the error was flash related, then immediately set the error counter to max
100                      (graceful abort). */
101                     else if (X_ERROR_FLASH == packet_status)
102                     {
103                         error_number = X_MAX_ERRORS;
104                         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
105                     }
106             }
107         }
108     }
109 }
```

```

103         /* Error while processing the packet, either send a NAK or do graceful abort. */
104     else
105     {
106         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
107     }
108     break;
109     /* End of Transmission. */
110 case X_EOT:
111     /* ACK, feedback to user (as a text), then jump to user application. */
112     Serial_PutByte(X_ACK);
113     Serial_PutString("\n\rFirmware updated!\n\r");
114
115     // Write out size of app
116     memory_extern->write(1, 0xffff04, xmodem_actual_flash_address);
117     uint8_t size_c[30];
118     sprintf((char*)size_c, "Write Out: %d\r\n", xmodem_actual_flash_address);
119     HAL_UART_Transmit(SERIAL_PC, size_c, strlen((char*)size_c), 10);
120     // Write out version of app
121     uint8_t ver[30];
122     sprintf((char*)ver, "VERSION: %s\r\n", VERSION);
123     HAL_UART_Transmit(SERIAL_PC, ver, strlen((char*)ver), 10);
124
125     memory_extern->write(1, 0xffff08, MCU_VERSION_MAJOR);
126     memory_extern->write(1, 0xffff0c, MCU_VERSION_MINOR);
127     memory_extern->write(1, 0xffff10, MCU_VERSION_PATCH);
128     memory_extern->write(1, 0xffff14, MCU_VERSION_RC);
129
130     Serial_PutString("Jumping to user application... \n\r");
131     HAL_Delay(2500);
132     NVIC_SystemReset();
133     break;
134     /* Abort from host. */
135 case X_CAN:
136     status = X_ERROR;
137     break;
138 default:
139     /* Wrong header. */
140     if (HAL_OK == comm_status)
141     {
142         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
143     }
144     break;
145 }
146 }
147 }
```

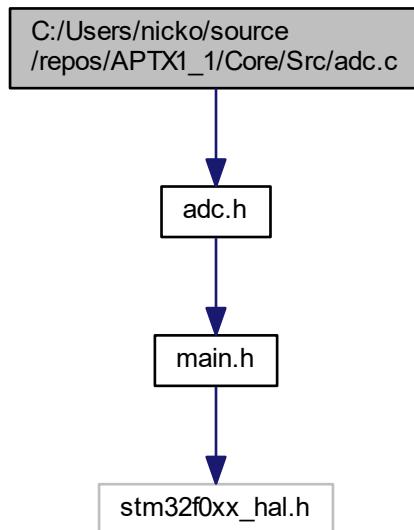
Here is the caller graph for this function:



6.28 C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc.c File Reference

```
#include "adc.h"
```

Include dependency graph for adc.c:



Functions

- void [MX_ADC_Init](#) (void)
- void [HAL_ADC_MspInit](#) (ADC_HandleTypeDef *adcHandle)
- void [HAL_ADC_MspDeInit](#) (ADC_HandleTypeDef *adcHandle)

Variables

- ADC_HandleTypeDef [hadc](#)
- DMA_HandleTypeDef [hdma_adc](#)

6.28.1 Function Documentation

6.28.1.1 HAL_ADC_MspDeInit()

```
void HAL_ADC_MspDeInit (
    ADC_HandleTypeDef * adcHandle )
```

ADC GPIO Configuration
PC0 ----> ADC_IN10

Definition at line 132 of file adc.c.

133 {

```

134
135     if (adcHandle->Instance==ADC1)
136     {
137         /* USER CODE BEGIN ADC1_MspDeInit_0 */
138
139         /* USER CODE END ADC1_MspDeInit_0 */
140         /* Peripheral clock disable */
141         __HAL_RCC_ADC1_CLK_DISABLE();
142
143         /**ADC GPIO Configuration
144          PC0      -----> ADC_IN10
145          */
146         HAL_GPIO_DeInit(ADC_CURRENT_SENSE_GPIO_Port, ADC_CURRENT_SENSE_Pin);
147
148         /* ADC1 DMA DeInit */
149         HAL_DMA_DeInit(adcHandle->DMA_Handle);
150     /* USER CODE BEGIN ADC1_MspDeInit_1 */
151
152     /* USER CODE END ADC1_MspDeInit_1 */
153 }
154 }
```

6.28.1.2 HAL_ADC_MspInit()

```
void HAL_ADC_MspInit (
    ADC_HandleTypeDef * adcHandle )
```

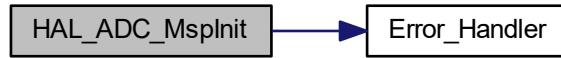
ADC GPIO Configuration
PC0 -----> ADC_IN10

Definition at line 88 of file adc.c.

```

89 {
90
91     GPIO_InitTypeDef GPIO_InitStruct = {0};
92     if (adcHandle->Instance==ADC1)
93     {
94         /* USER CODE BEGIN ADC1_MspInit_0 */
95
96         /* USER CODE END ADC1_MspInit_0 */
97         /* ADC1 clock enable */
98         __HAL_RCC_ADC1_CLK_ENABLE();
99
100        __HAL_RCC_GPIOC_CLK_ENABLE();
101        /**ADC GPIO Configuration
102          PC0      -----> ADC_IN10
103          */
104        GPIO_InitStruct.Pin = ADC_CURRENT_SENSE_Pin;
105        GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
106        GPIO_InitStruct.Pull = GPIO_NOPULL;
107        HAL_GPIO_Init(ADC_CURRENT_SENSE_GPIO_Port, &GPIO_InitStruct);
108
109        /* ADC1 DMA Init */
110        /* ADC Init */
111        hdma_adc.Instance = DMA1_Channel1;
112        hdma_adc.Init.Direction = DMA_PERIPH_TO_MEMORY;
113        hdma_adc.InitPeriphInc = DMA_PINC_DISABLE;
114        hdma_adc.InitMemInc = DMA_MINC_ENABLE;
115        hdma_adc.InitPeriphDataAlignment = DMA_PDATAALIGN_WORD;
116        hdma_adc.InitMemDataAlignment = DMA_MDATAALIGN_WORD;
117        hdma_adc.Init.Mode = DMA_CIRCULAR;
118        hdma_adc.Init.Priority = DMA_PRIORITY_LOW;
119        if (HAL_DMA_Init(&hdma_adc) != HAL_OK)
120        {
121            Error_Handler();
122        }
123
124        __HAL_LINKDMA(adcHandle,DMA_Handle,hdma_adc);
125
126     /* USER CODE BEGIN ADC1_MspInit_1 */
127
128     /* USER CODE END ADC1_MspInit_1 */
129 }
130 }
```

Here is the call graph for this function:



6.28.1.3 MX_ADC_Init()

```
void MX_ADC_Init (
    void )
```

Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)

Configure for the selected ADC regular channel to be converted.

Configure for the selected ADC regular channel to be converted.

Configure for the selected ADC regular channel to be converted.

Configure for the selected ADC regular channel to be converted.

Definition at line 31 of file adc.c.

```
32 {
33     ADC_ChannelConfTypeDef sConfig = {0};
34
35     /* Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of
       conversion)
36 */
37     hadc.Instance = ADC1;
38     hadc.Init.ClockPrescaler = ADC_CLOCK_ASYNC_DIV1;
39     hadc.Init.Resolution = ADC_RESOLUTION_10B;
40     hadc.Init.DataAlign = ADC_DATAALIGN_RIGHT;
41     hadc.Init.ScanConvMode = ADC_SCAN_DIRECTION_FORWARD;
42     hadc.Init.EOCSelection = ADC_EOC_SEQ_CONV;
43     hadc.Init.LowPowerAutoWait = DISABLE;
44     hadc.Init.LowPowerAutoPowerOff = DISABLE;
45     hadc.Init.ContinuousConvMode = ENABLE;
46     hadc.Init.DiscontinuousConvMode = DISABLE;
47     hadc.Init.ExternalTrigConv = ADC_SOFTWARE_START;
48     hadc.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
49     hadc.Init.DMAContinuousRequests = ENABLE;
50     hadc.Init.Overrun = ADC_OVR_DATA_OVERWRITTEN;
51     if (HAL_ADC_Init(&hadc) != HAL_OK)
52     {
53         Error_Handler();
54     }
55     /* Configure for the selected ADC regular channel to be converted.
56 */
57     sConfig.Channel = ADC_CHANNEL_10;
58     sConfig.Rank = ADC_RANK_CHANNEL_NUMBER;
59     sConfig.SamplingTime = ADC_SAMPLETIME_239CYCLES_5;
60     if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
61     {
62         Error_Handler();
63     }
64     /* Configure for the selected ADC regular channel to be converted.
65 */
66     sConfig.Channel = ADC_CHANNEL_TEMPSENSOR;
67     if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
68     {
```

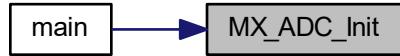
```

69     Error_Handler();
70 }
71 /** Configure for the selected ADC regular channel to be converted.
72 */
73 sConfig.Channel = ADC_CHANNEL_VREFINT;
74 if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
75 {
76     Error_Handler();
77 }
78 /** Configure for the selected ADC regular channel to be converted.
79 */
80 sConfig.Channel = ADC_CHANNEL_VBAT;
81 if (HAL_ADC_ConfigChannel(&hadc, &sConfig) != HAL_OK)
82 {
83     Error_Handler();
84 }
85
86 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.28.2 Variable Documentation

6.28.2.1 hadc

ADC_HandleTypeDef hadc

File Name : [ADC.c](#) Description : This file provides code for the configuration of the ADC instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file adc.c.

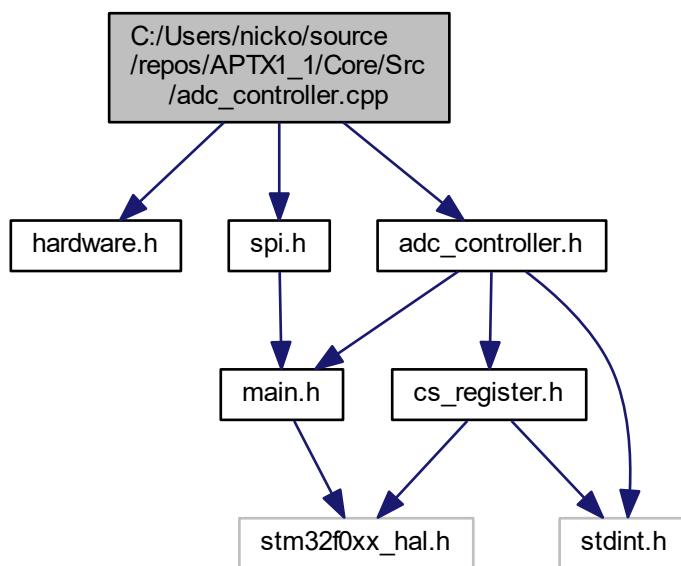
6.28.2.2 hdma_adc

DMA_HandleTypeDef hdma_adc

Definition at line 28 of file adc.c.

6.29 C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc_controller.cpp File Reference

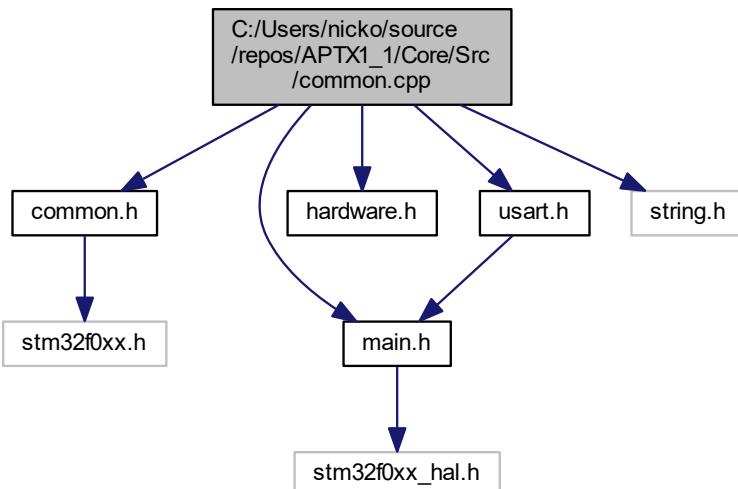
```
#include "hardware.h"
#include "adc_controller.h"
#include "spi.h"
Include dependency graph for adc_controller.cpp:
```



6.30 C:/Users/nicko/source/repos/APTX1_1/Core/Src/common.cpp File Reference

This file provides all the common functions.

```
#include "common.h"
#include "main.h"
#include "hardware.h"
#include "uart.h"
#include "string.h"
Include dependency graph for common.cpp:
```



Functions

- void [Int2Str](#) (uint8_t *p_str, uint32_t intnum)
Convert an Integer to a string.
- uint32_t [Str2Int](#) (uint8_t *p_inputstr, uint32_t *p_intnum)
Convert a string to an integer.
- void [Serial_PutString](#) (const char *p_string)
Print a string on the HyperTerminal.
- HAL_StatusTypeDef [Serial_PutByte](#) (uint8_t param)
Transmit a byte to the HyperTerminal.

6.30.1 Detailed Description

This file provides all the common functions.

Author

MCD Application Team

Version

1.0.0

Date

8-April-2015

Attention

© COPYRIGHT(c) 2015 STMicroelectronics

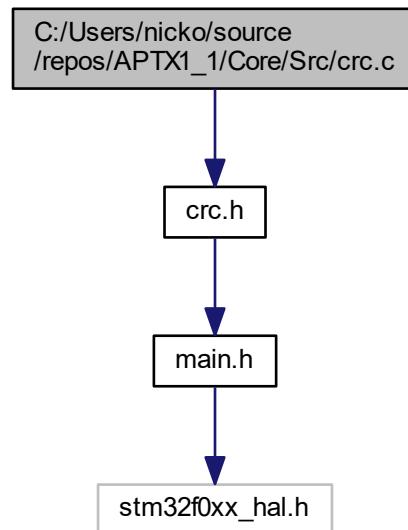
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.31 C:/Users/nicko/source/repos/APTX1_1/Core/Src/crc.c File Reference

```
#include "crc.h"
Include dependency graph for crc.c:
```



Functions

- void [MX_CRC_Init](#) (void)
- void [HAL_CRC_MspInit](#) (CRC_HandleTypeDef *crcHandle)
- void [HAL_CRC_MspDelInit](#) (CRC_HandleTypeDef *crcHandle)

Variables

- CRC_HandleTypeDef [hcrc](#)

6.31.1 Function Documentation

6.31.1.1 HAL_CRC_MspDeInit()

```
void HAL_CRC_MspDeInit (
    CRC_HandleTypeDef * crcHandle )
```

Definition at line 62 of file crc.c.

```
63 {
64
65     if(crcHandle->Instance==CRC)
66     {
67         /* USER CODE BEGIN CRC_MspDeInit 0 */
68
69         /* USER CODE END CRC_MspDeInit 0 */
70         /* Peripheral clock disable */
71         __HAL_RCC_CRC_CLK_DISABLE();
72
73         /* USER CODE BEGIN CRC_MspDeInit 1 */
74
75     }
76 }
```

6.31.1.2 HAL_CRC_MspInit()

```
void HAL_CRC_MspInit (
    CRC_HandleTypeDef * crcHandle )
```

Definition at line 46 of file crc.c.

```
47 {
48
49     if(crcHandle->Instance==CRC)
50     {
51         /* USER CODE BEGIN CRC_MspInit 0 */
52
53         /* USER CODE END CRC_MspInit 0 */
54         /* CRC clock enable */
55         __HAL_RCC_CRC_CLK_ENABLE();
56
57         /* USER CODE BEGIN CRC_MspInit 1 */
58
59     }
60 }
```

6.31.1.3 MX_CRC_Init()

```
void MX_CRC_Init (
    void )
```

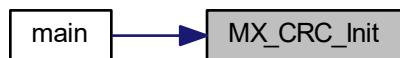
Definition at line 30 of file crc.c.

```
31 {
32
33     hcrc.Instance = CRC;
34     hcrc.Init.DefaultPolynomialUse = DEFAULT_POLYNOMIAL_ENABLE;
35     hcrc.Init.DefaultInitValueUse = DEFAULT_INIT_VALUE_ENABLE;
36     hcrc.Init.InputDataInversionMode = CRC_INPUTDATA_INVERSION_NONE;
37     hcrc.Init.OutputDataInversionMode = CRC_OUTPUTDATA_INVERSION_DISABLE;
38     hcrc.InputDateFormat = CRC_INPUTDATA_FORMAT_BYTES;
39     if (HAL_CRC_Init(&hcrc) != HAL_OK)
40     {
41         Error_Handler();
42     }
43
44 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.2 Variable Documentation

6.31.2.1 hcrc

CRC_HandleTypeDef hcrc

File Name : [CRC.c](#) Description : This file provides code for the configuration of the CRC instances.

Attention

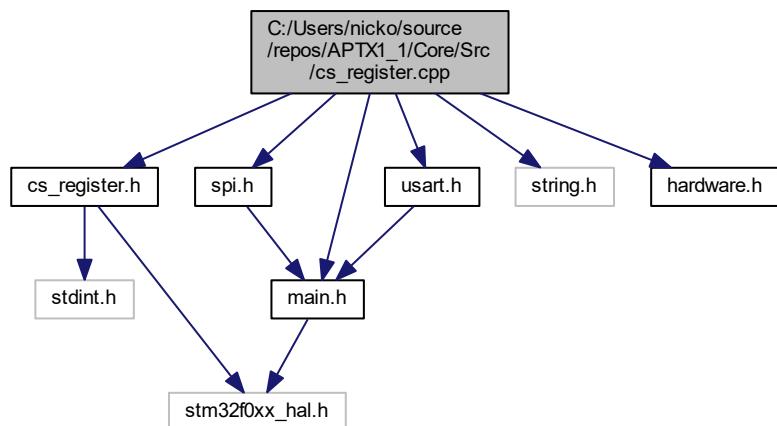
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file `crc.c`.

6.32 C:/Users/nicko/source/repos/APTX1_1/Core/Src/cs_register.cpp File Reference

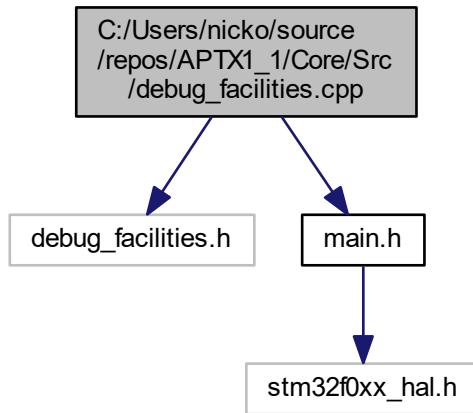
```
#include "cs_register.h"
#include "spi.h"
#include "main.h"
#include "uart.h"
#include <string.h>
#include "hardware.h"
Include dependency graph for cs_register.cpp:
```



6.33 C:/Users/nicko/source/repos/APTX1_1/Core/Src/debug_facilities.cpp File Reference

```
#include "debug_facilities.h"
#include "main.h"
```

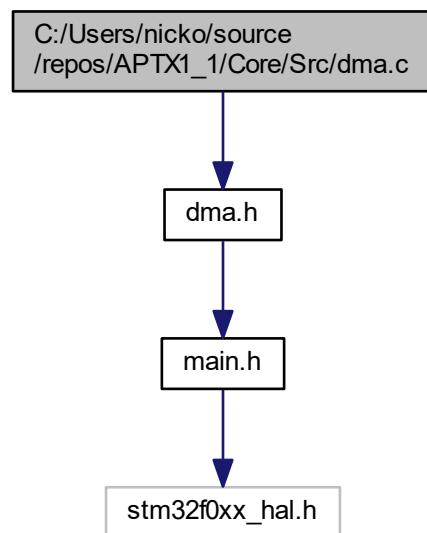
Include dependency graph for debug_facilities.cpp:



6.34 C:/Users/nicko/source/repos/APTX1_1/Core/Src/dma.c File Reference

```
#include "dma.h"
```

Include dependency graph for dma.c:



Functions

- void [MX_DMA_Init](#) (void)

6.34.1 Function Documentation

6.34.1.1 MX_DMA_Init()

```
void MX_DMA_Init (
    void )
```

File Name : [dma.c](#) Description : This file provides code for the configuration of all the requested memory to memory DMA transfers.

Attention

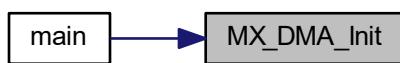
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 38 of file dma.c.

```
39 {
40
41     /* DMA controller clock enable */
42     __HAL_RCC_DMA1_CLK_ENABLE();
43
44     /* DMA interrupt init */
45     /* DMA1_Channel1_IRQHandler interrupt configuration */
46     HAL_NVIC_SetPriority(DMA1_Channel1_IRQHandler, 0, 0);
47     HAL_NVIC_EnableIRQ(DMA1_Channel1_IRQHandler);
48
49 }
```

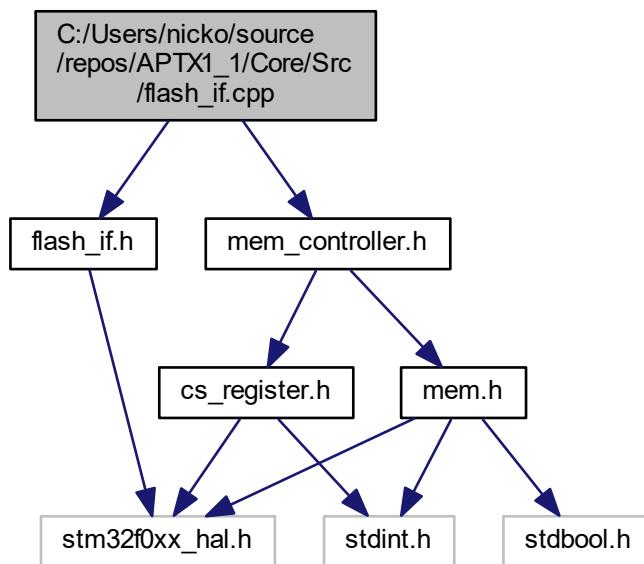
Here is the caller graph for this function:



6.35 C:/Users/nicko/source/repos/APTX1_1/Core/Src/flash_if.cpp File Reference

This file provides all the memory related operation functions.

```
#include "flash_if.h"
#include "mem_controller.h"
Include dependency graph for flash_if.cpp:
```



Functions

- void [FLASH_If_Erase](#) (void)
This function does an erase of all user flash area.
- uint32_t [FLASH_If_Write](#) (uint32_t destination, uint8_t *p_source, uint32_t length)
This function writes a data buffer in flash (data are 32-bit aligned).

Variables

- [Mem_ctrl * memory_extern](#)

6.35.1 Detailed Description

This file provides all the memory related operation functions.

Author

MCD Application Team

Version

1.0.0

Date

8-April-2015

Attention

© COPYRIGHT(c) 2015 STMicroelectronics

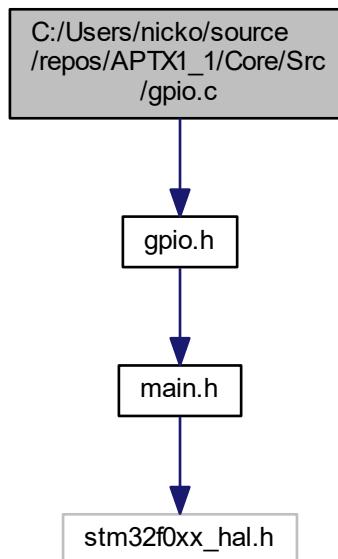
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.36 C:/Users/nicko/source/repos/APTX1_1/Core/Src/gpio.c File Reference

```
#include "gpio.h"
Include dependency graph for gpio.c:
```



Functions

- void [MX_GPIO_Init](#) (void)

6.36.1 Function Documentation

6.36.1.1 MX_GPIO_Init()

```
void MX_GPIO_Init (
    void
)
```

File Name : [gpio.c](#) Description : This file provides code for the configuration of all used GPIO pins.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause Configure pins as Analog Input Output EVENT_OUT EXTI PA8 ----> RCC_MCO

Definition at line 41 of file gpio.c.

```

42 {
43
44     GPIO_InitTypeDef GPIO_InitStruct = {0};
45
46     /* GPIO Ports Clock Enable */
47     __HAL_RCC_GPIOC_CLK_ENABLE();
48     __HAL_RCC_GPIOF_CLK_ENABLE();
49     __HAL_RCC_GPIOA_CLK_ENABLE();
50     __HAL_RCC_GPIOB_CLK_ENABLE();
51     __HAL_RCC_GPIOD_CLK_ENABLE();
52
53     /*Configure GPIO pin Output Level */
54     HAL_GPIO_WritePin(GPIOC,
55                         ENABLE_POWER_12V_Pin|LVDS_RESERVED_Pin|CHIP_SELECT_RESERVED_Pin|RESERVED_UART6_TX_Pin
56                         |RESERVED_UART6_RX_Pin|DEBUG_MOTOR_RUNNING_Pin|DEBUG_PRESSURE_OK_Pin,
57                         GPIO_PIN_RESET);
58
59     /*Configure GPIO pin Output Level */
60     HAL_GPIO_WritePin(GPIOA, CHIP_SELECT_SERIAL_Pin|USB_DM_RESERVED_Pin|USB_DP_RESERVED_Pin,
61                         GPIO_PIN_RESET);
62
63     /*Configure GPIO pin Output Level */
64     HAL_GPIO_WritePin(GPIOB, CHIP_SELECT MCU_MEM_Pin|WATCHDOG_OUT_Pin|DEBUG_CLK_AUX_Pin|BUZZER_Pin
65                         |BLINKING_LED_Pin|DEBUG_AM_OK_Pin|DEBUG_APT_OK_Pin|DEBUG MCU_OK_Pin
66                         |SYS_SWO_RESERVED_Pin|DEBUG_DATA_AUX_Pin|DEBUG_STATE_3_Pin|DEBUG_STATE_2_Pin
67                         |DEBUG_STATE_1_Pin|DEBUG_BRK1_Pin|DEBUG_BRK2_Pin, GPIO_PIN_RESET);
68
69     /*Configure GPIO pins : PCPin PCPin PCPin PCPin
70                         PCPin */
71     GPIO_InitStruct.Pin =
72         TOGGLE400_RESET_BTN_Pin|SIMPLE_INTERLOCK_Pin|LOGIC_BYPASS_3_Pin|LOGIC_BYPASS_2_Pin
73         |LOGIC_BYPASS_1_Pin;
74     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
75     GPIO_InitStruct.Pull = GPIO_NOPULL;
76     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
77
78     /*Configure GPIO pins : PCPin PCPin PCPin PCPin
79                         PCPin PCPin PCPin */
80     GPIO_InitStruct.Pin =
81         ENABLE_POWER_12V_Pin|LVDS_RESERVED_Pin|CHIP_SELECT_RESERVED_Pin|RESERVED_UART6_TX_Pin
82         |RESERVED_UART6_RX_Pin|DEBUG_MOTOR_RUNNING_Pin|DEBUG_PRESSURE_OK_Pin;
83     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
84     GPIO_InitStruct.Pull = GPIO_NOPULL;
85     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
86     HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
87
88     /*Configure GPIO pins : PAPin PAPin */
89     GPIO_InitStruct.Pin = PA1_RESERVED_ASK_SHAUL_Pin|LOGIC_BYPASS_4_Pin;
90     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
91     GPIO_InitStruct.Pull = GPIO_NOPULL;
92     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
93
94     /*Configure GPIO pins : PAPin PAPin PAPin */
95     GPIO_InitStruct.Pin = CHIP_SELECT_SERIAL_Pin|USB_DM_RESERVED_Pin|USB_DP_RESERVED_Pin;
96     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
97     GPIO_InitStruct.Pull = GPIO_NOPULL;
98     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
99     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
100
101    /*Configure GPIO pins : PBPin PBPin PBPin PBPin
102                         PBPin PBPin PBPin PBPin
103                         PBPin PBPin PBPin */
104    GPIO_InitStruct.Pin = CHIP_SELECT MCU_MEM_Pin|WATCHDOG_OUT_Pin|DEBUG_CLK_AUX_Pin|BUZZER_Pin
105                         |BLINKING_LED_Pin|DEBUG_AM_OK_Pin|DEBUG_APT_OK_Pin|DEBUG MCU_OK_Pin
106                         |SYS_SWO_RESERVED_Pin|DEBUG_DATA_AUX_Pin|DEBUG_STATE_3_Pin|DEBUG_STATE_2_Pin
107                         |DEBUG_STATE_1_Pin|DEBUG_BRK1_Pin|DEBUG_BRK2_Pin;
108
109    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;

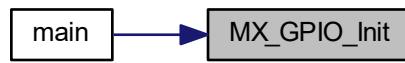
```

```

108     GPIO_InitStruct.Pull = GPIO_NOPULL;
109     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
110     HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);
111
112     /*Configure GPIO pin : PtPin */
113     GPIO_InitStruct.Pin = POWER_FAIL MCU_Pin;
114     GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
115     GPIO_InitStruct.Pull = GPIO_NOPULL;
116     HAL_GPIO_Init(POWER_FAIL MCU_GPIO_Port, &GPIO_InitStruct);
117
118     /*Configure GPIO pin : PA8 */
119     GPIO_InitStruct.Pin = GPIO_PIN_8;
120     GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
121     GPIO_InitStruct.Pull = GPIO_NOPULL;
122     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
123     GPIO_InitStruct.Alternate = GPIO_AF0_MCO;
124     HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
125
126     /*Configure GPIO pin : PtPin */
127     GPIO_InitStruct.Pin = DEBUG_LATCH_AUX_Pin;
128     GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
129     GPIO_InitStruct.Pull = GPIO_NOPULL;
130     GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
131     HAL_GPIO_Init(DEBUG_LATCH_AUX_GPIO_Port, &GPIO_InitStruct);
132
133 }

```

Here is the caller graph for this function:



6.37 C:/Users/nicko/source/repos/APTX1_1/Core/Src/main.cpp File Reference

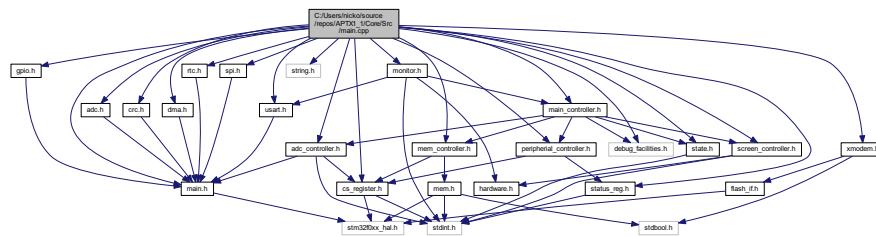
: Main program body

```

#include "main.h"
#include "adc.h"
#include "crc.h"
#include "dma.h"
#include "rtc.h"
#include "spi.h"
#include "usart.h"
#include "gpio.h"
#include "string.h"
#include "cs_register.h"
#include "status_reg.h"
#include "mem_controller.h"
#include "screen_controller.h"
#include "monitor.h"
#include "state.h"
#include "debug_facilities.h"
#include "peripheral_controller.h"
#include "adc_controller.h"
#include "main_controller.h"

```

```
#include "xmodem.h"
Include dependency graph for main.cpp:
```



Macros

- `#define __HAL_SYSCFG_REMAPMEMORY_SRAM()`

Functions

- `void SystemClock_Config (void)`
System Clock Configuration.
- `void HAL_ADC_ConvCpltCallback (ADC_HandleTypeDef *hadc)`
- `void remapMemToSRAM (void)`
- `volatile uint32_t VectorTable[48] __attribute__ ((section(".RAMVectorTable")))`
- `int main (void)`
The application entry point.
- `void HAL_TIM_PeriodElapsedCallback (TIM_HandleTypeDef *htim)`
Period elapsed callback in non blocking mode.
- `void Error_Handler (void)`
This function is executed in case of error occurrence.

Variables

- `DebugHelper debug = DebugHelper()`
- `CS_reg cs_register = CS_reg()`
- `STATUS_reg status_register = STATUS_reg()`
- `Memory memory_am = Memory(3)`
- `Memory memory_apt = Memory(2)`
- `Memory memory_mcu = Memory(1)`
- `Mem_ctrl memory_controller = Mem_ctrl(&memory_am, &memory_apt, &memory_mcu, &cs_register)`
- `PeripheralDevices ph_device = PeripheralDevices(&cs_register, &status_register)`
- `State state = State()`
- `Screen_ctrl screen = Screen_ctrl()`
- `AdcController adc_controller = AdcController(&cs_register)`
- `uint32_t cycle = 0`
- `MainController grand_ctrl = MainController(&debug, &memory_controller, &ph_device, &state, &screen, &adc_controller)`
- `MainController * atp_extern = &grand_ctrl`
- `Mem_ctrl * memory_extern = &memory_controller`
- `uint8_t aPacketData [PACKET_1K_SIZE+PACKET_DATA_INDEX+PACKET_TRAILER_SIZE]`
- `bool reset_pressed = false`
- `uint32_t timer_last_update`

6.37.1 Detailed Description

: Main program body

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

6.37.2 Macro Definition Documentation

6.37.2.1 __HAL_SYSCFG_REMAPMEMORY_SRAM

```
#define __HAL_SYSCFG_REMAPMEMORY_SRAM( )
```

Value:

```
do {SYSCFG->CFGREG1 &= (SYSCFG_CFGREG1_MEM_MODE); \
    SYSCFG->CFGREG1 |= (SYSCFG_CFGREG1_MEM_MODE_0 | SYSCFG_CFGREG1_MEM_MODE_1); \
} while(0)
```

Definition at line 59 of file main.cpp.

6.37.3 Function Documentation

6.37.3.1 __attribute__()

```
volatile uint32_t VectorTable [48] __attribute__ (
    section(".RAMVectorTable"))
```

6.37.3.2 Error_Handler()

```
void Error_Handler (
    void )
```

This function is executed in case of error occurrence.

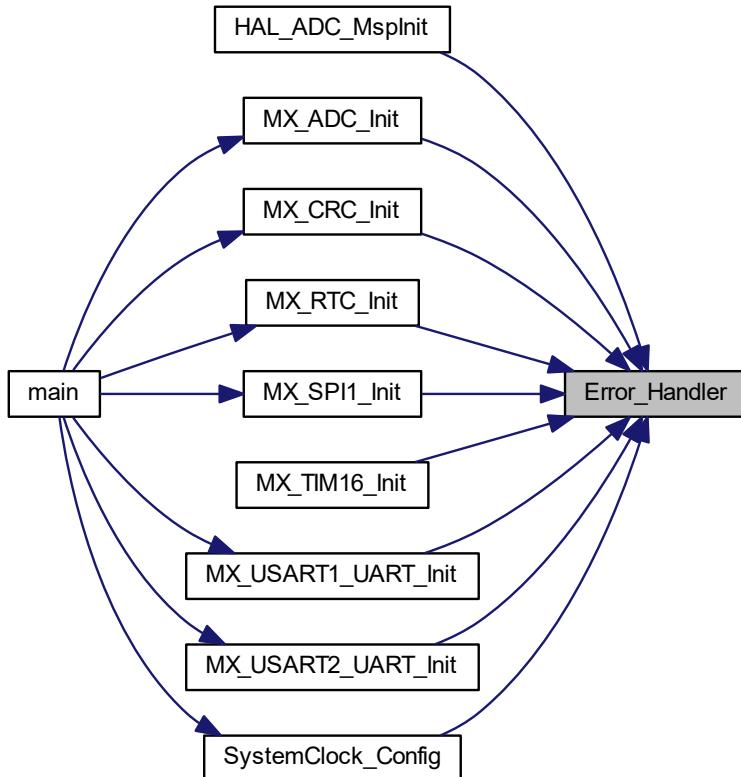
Return values

`None`

Definition at line 323 of file main.cpp.

```
325 {  
326     /* USER CODE BEGIN Error_Handler_Debug */  
327     /* User can add his own implementation to report the HAL error return state */  
328  
329     /* USER CODE END Error_Handler_Debug */
```

Here is the caller graph for this function:



6.37.3.3 HAL_ADC_ConvCpltCallback()

```
void HAL_ADC_ConvCpltCallback (  
    ADC_HandleTypeDef * hadc )
```

Definition at line 98 of file main.cpp.

```
99 {  
100     adc_controller.current_sense  = adc_controller.dma_ring_buffer[0];  
101     adc_controller.temperature_mcu = adc_controller.dma_ring_buffer[0];  
102     adc_controller.vref   = adc_controller.dma_ring_buffer[0];  
103     adc_controller.vbat   = adc_controller.dma_ring_buffer[0];  
104 }
```

6.37.3.4 HAL_TIM_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
    TIM_HandleTypeDef * htim )
```

Period elapsed callback in non blocking mode.

Note

This function is called when TIM1 interrupt took place, inside HAL_TIM_IRQHandler(). It makes a direct call to HAL_IncTick() to increment a global variable "uwTick" used as application time base.

Parameters

<i>htim</i>	: TIM handle
-------------	--------------

Return values

<i>None</i>	
-------------	--

Definition at line 306 of file main.cpp.

```
308 {
309     /* USER CODE BEGIN Callback 0 */
310
311     /* USER CODE END Callback 0 */
312     if (htim->Instance == TIM1) {
313         HAL_IncTick();
314     }
315     /* USER CODE BEGIN Callback 1 */
316
317     /* USER CODE END Callback 1 */
```

6.37.3.5 main()

```
int main (
    void )
```

The application entry point.

Return values

<i>int</i>	
------------	--

26.03.20: alpha release 1: changelog: So far we have all peripherals working together to create the basic flow we can read all memories, query adc, and get status. 29.03.20 alphaR2: add a main class to abstract the different classes add qa reduced functionality add timer on error messages for tx_send_general error 30.03.20: add xmodem update/upload add rx_routines class to instrument functions from uart 01/04/20: add TOGGLE_RESET_BUTTON read add stalling procedures FIX: adc design read back cs register function:

1. change CS_REG status
2. change STATUS_REG When cs_low at cs_register, devices online will be listening to the cs_register setup commands.

Definition at line 132 of file main.cpp.

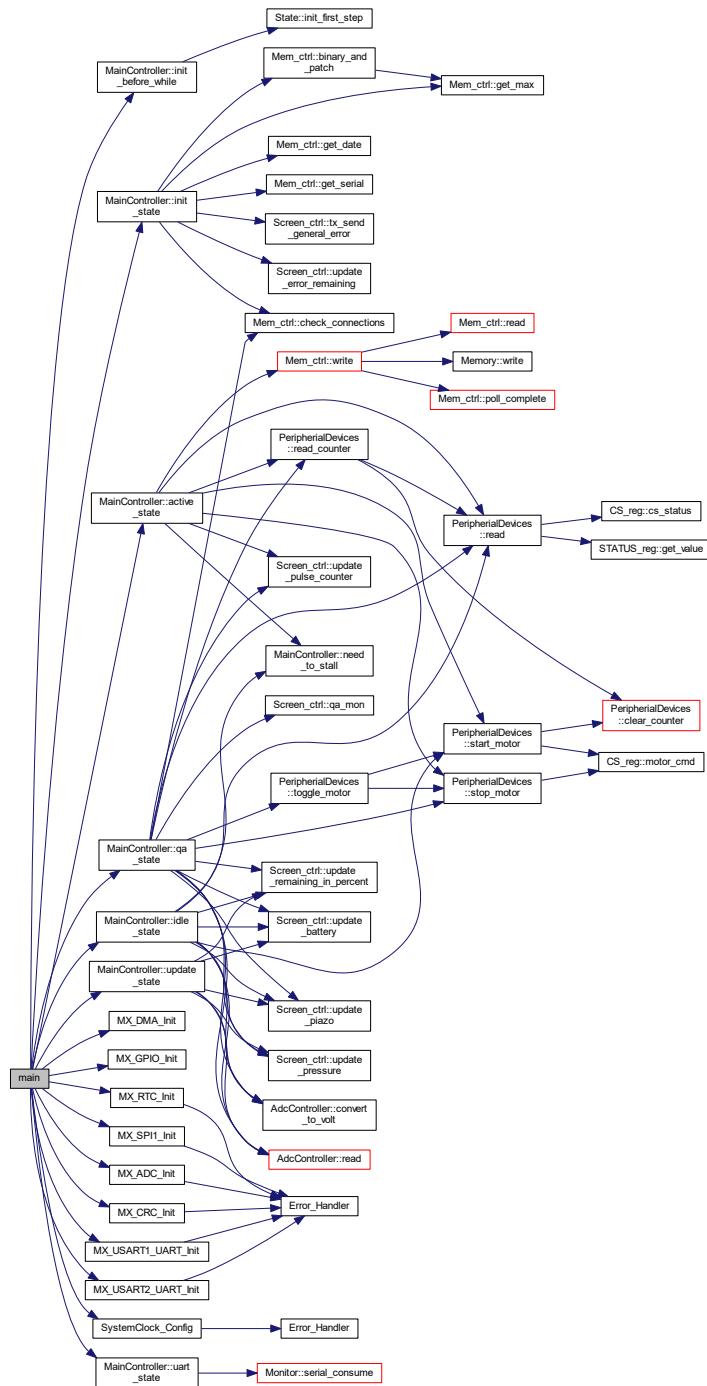
```

133 {
134     /* USER CODE BEGIN 1 */
135
136     /* USER CODE END 1 */
137
138
139     /* MCU Configuration-----*/
140
141     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
142     HAL_Init();
143
144     /* USER CODE BEGIN Init */
145
146     /* USER CODE END Init */
147
148     /* Configure the system clock */
149     SystemClock_Config();
150
151     /* USER CODE BEGIN SysInit */
152
153     /* USER CODE END SysInit */
154
155     /* Initialize all configured peripherals */
156     MX_GPIO_Init();
157     MX_DMA_Init();
158     MX_ADC_Init();
159     MX_RTC_Init();
160     MX_SPI1_Init();
161     MX_USART1_UART_Init();
162     MX_CRC_Init();
163     MX_USART2_UART_Init();
164     /* USER CODE BEGIN 2 */
165     HAL_ADC_Start_DMA(&hadc, adc_controller.dma_ring_buffer, DMA_LENGTH);
166     grand_ctrl.init_before_while();
167     timer_last_update = HAL_GetTick();
168     /* USER CODE END 2 */
169
170
171     /* Infinite loop */
172     /* USER CODE BEGIN WHILE */
173     while (true)
174     {
175         //*****
176         *26.03.20: alpha release 1:
177         *changelog:
178         *So far we have all peripherals working together to create the basic flow
179         *we can read all memories, query adc, and get status.
180         *29.03.20 alphaR2:
181         *add a main class to abstract the different classes
182         *add qa reduced functionality
183         *add timer on error messages for tx_send_general_error
184         *30.03.20:
185         *add xmodem update/upload
186         *add rx_routines class to instrument functions from uart
187         *01/04/20:
188         *add TOGGLE_RESET_BUTTON read
189         *add stalling procedures
190         *FIX:
191         *adc design
192         *read back cs register function:
193         * 1. change CS_REG status
194         * 2. change STATUS_REG
195         *When cs_low at cs_register, devices online will be listening to the cs_register setup commands.
196         ****
197
198         switch (state.current)
199         {
200             case Init:
201                 grand_ctrl.init_state();
202                 break;
203             case Idle:
204                 grand_ctrl.idle_state();
205                 break;
206             case UpdateData:
207                 grand_ctrl.update_state();
208                 break;
209             case Active:
210                 grand_ctrl.active_state();
211                 break;
212             case UartDebug:
213                 grand_ctrl.uart_state();
214                 break;
215             case QA:
216                 grand_ctrl.qa_state();
217                 break;
218             default :

```

```
219         break;
220     }
221     if (HAL_GetTick() - timer_last_update > 40)
222     {
223         state.current = UpdateData;
224         timer_last_update = HAL_GetTick();
225     }
226     else
227     {
228         state.current = state.next;
229     }
230
231     HAL_GPIO_TogglePin(WATCHDOG_OUT_GPIO_Port, WATCHDOG_OUT_Pin);
232     cycle++;
233     if (cycle % 100 == 0)
234     {
235         HAL_GPIO_TogglePin(BLINKING_LED_GPIO_Port, BLINKING_LED_Pin);
236     }
237 /* USER CODE END WHILE */
238
239 /* USER CODE BEGIN 3 */
240 }
241 /* USER CODE END 3 */
```

Here is the call graph for this function:



6.37.3.6 remapMemToSRAM()

```
void remapMemToSRAM (
    void )
```

Definition at line 106 of file main.cpp.

```

107 {
108     uint32_t vecIndex = 0;
109     uint32_t prim;
110     prim = __get_PRIMASK();
111
112     __disable_irq();
113
114     for (vecIndex = 0; vecIndex < 48; vecIndex++) {
115         VectorTable[vecIndex] = *(volatile uint32_t*)(APP_ADDRESS + (vecIndex << 2));
116     }
117
118     //__HAL_RCC_APB2_FORCE_RESET();
119     __HAL_SYSCFG_REMAPMEMORY_SRAM();
120
121     if (!prim) {
122         __enable_irq();
123     }
124 }
```

6.37.3.7 SystemClock_Config()

```
void SystemClock_Config (
    void )
```

System Clock Configuration.

Return values

None	
------	--

Configure LSE Drive Capability

Initializes the CPU, AHB and APB busses clocks

Initializes the CPU, AHB and APB busses clocks

Definition at line 247 of file main.cpp.

```

249 {
250     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
251     RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
252     RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
253
254     /** Configure LSE Drive Capability
255     */
256     HAL_PWR_EnableBkUpAccess();
257     __HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_HIGH);
258     /** Initializes the CPU, AHB and APB busses clocks
259     */
260     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI48|RCC_OSCILLATORTYPE_LSE;
261     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
262     RCC_OscInitStruct.LSEState = RCC_LSE_ON;
263     RCC_OscInitStruct.HSI48State = RCC_HSI48_ON;
264     RCC_OscInitStruct.HSI14State = RCC_HSI14_ON;
265     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
266     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
267     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
268     {
269         Error_Handler();
270     }
271     /** Initializes the CPU, AHB and APB busses clocks
272     */
273     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
274             |RCC_CLOCKTYPE_PCLK1;
275     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI48;
276     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
277     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
278
279     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
280     {
```

```

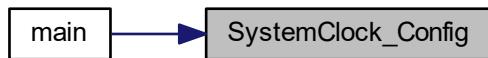
281     Error_Handler();
282 }
283 PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_USART2
284     |RCC_PERIPHCLK_RTC;
285 PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_SYSCLK;
286 PeriphClkInit.Usart2ClockSelection = RCC_USART2CLKSOURCE_PCLK1;
287 PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_LSE;
288 if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
289 {
290     Error_Handler();
291 }
292 HAL_RCC_MCOConfig(RCC_MCO, RCC_MCO1SOURCE_SYSCLK, RCC_MCODIV_1);

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.37.4 Variable Documentation

6.37.4.1 adc_controller

```
AdcController adc_controller = AdcController(&cs_register)
```

Definition at line 78 of file main.cpp.

6.37.4.2 aPacketData

```
uint8_t aPacketData[PACKET_1K_SIZE+PACKET_DATA_INDEX+PACKET_TRAILER_SIZE]
```

Definition at line 83 of file main.cpp.

6.37.4.3 atp_extern

```
MainController* atp_extern = &grand_ctrl
```

Definition at line 81 of file main.cpp.

6.37.4.4 cs_register

```
CS_Reg cs_register = CS_Reg()
```

Definition at line 69 of file main.cpp.

6.37.4.5 cycle

```
uint32_t cycle = 0
```

Definition at line 79 of file main.cpp.

6.37.4.6 debug

```
DebugHelper debug = DebugHelper()
```

Definition at line 68 of file main.cpp.

6.37.4.7 grand_ctrl

```
MainController grand_ctrl = MainController(&debug, &memory_controller, &ph_device, &state,  
&screen, &adc_controller)
```

Definition at line 80 of file main.cpp.

6.37.4.8 memory_am

```
Memory memory_am = Memory(3)
```

Definition at line 71 of file main.cpp.

6.37.4.9 memory_apt

```
Memory memory_apt = Memory(2)
```

Definition at line 72 of file main.cpp.

6.37.4.10 memory_controller

```
Mem_ctrl memory_controller = Mem_ctrl(&memory_am, &memory_apt, &memory_mcu, &cs_register)
```

Definition at line 74 of file main.cpp.

6.37.4.11 memory_mcu

```
Memory memory_mcu = Memory(1)
```

Definition at line 73 of file main.cpp.

6.37.4.12 ph_device

```
PeripheralDevices ph_device = PeripheralDevices(&cs_register, &status_register)
```

Definition at line 75 of file main.cpp.

6.37.4.13 reset_pressed

```
bool reset_pressed = false
```

Definition at line 84 of file main.cpp.

6.37.4.14 screen

```
Screen_ctrl screen = Screen_ctrl()
```

Definition at line 77 of file main.cpp.

6.37.4.15 state

```
State state = State()
```

Definition at line 76 of file main.cpp.

6.37.4.16 status_register

```
STATUS_Reg status_register = STATUS_Reg()
```

Definition at line 70 of file main.cpp.

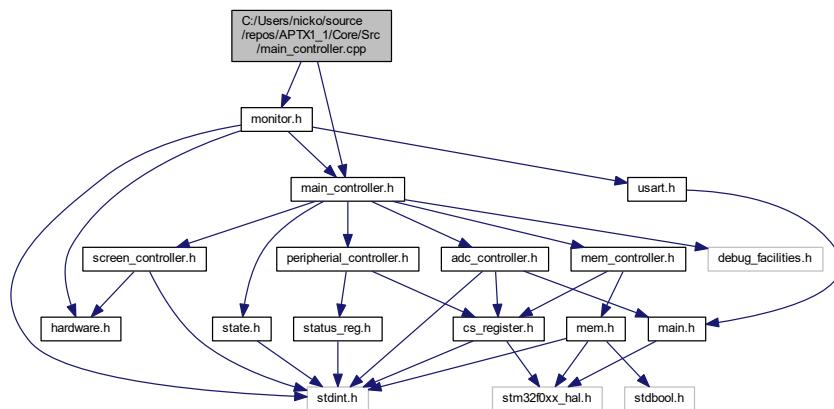
6.37.4.17 timer_last_update

```
uint32_t timer_last_update
```

Definition at line 85 of file main.cpp.

6.38 C:/Users/nicko/source/repos/APTX1_1/Core/Src/main_controller.cpp File Reference

```
#include "main_controller.h"
#include "monitor.h"
Include dependency graph for main_controller.cpp:
```



Variables

- bool reset_pressed

6.38.1 Variable Documentation

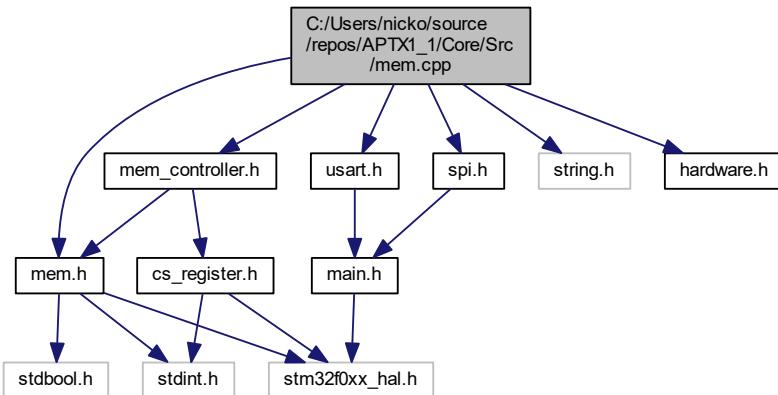
6.38.1.1 reset_pressed

```
bool reset_pressed
```

Definition at line 84 of file main.cpp.

6.39 C:/Users/nicko/source/repos/APTX1_1/Core/Src/mem.cpp File Reference

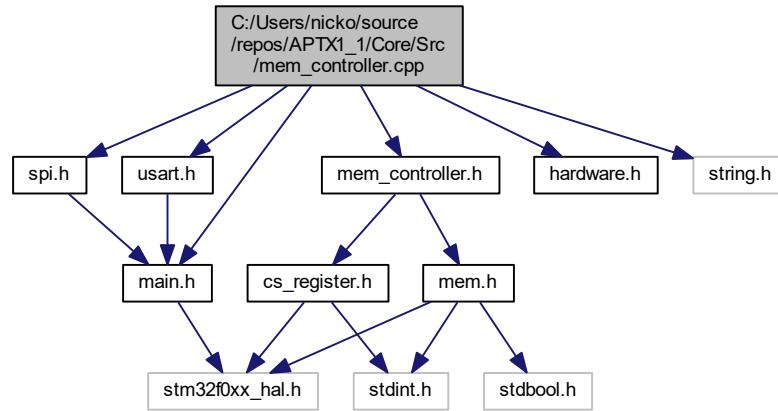
```
#include "mem.h"
#include "usart.h"
#include <string.h>
#include "hardware.h"
#include "mem_controller.h"
#include "spi.h"
Include dependency graph for mem.cpp:
```



6.40 C:/Users/nicko/source/repos/APTX1_1/Core/Src/mem_controller.cpp File Reference

```
#include "mem_controller.h"
#include "hardware.h"
#include "main.h"
#include "spi.h"
#include <string.h>
```

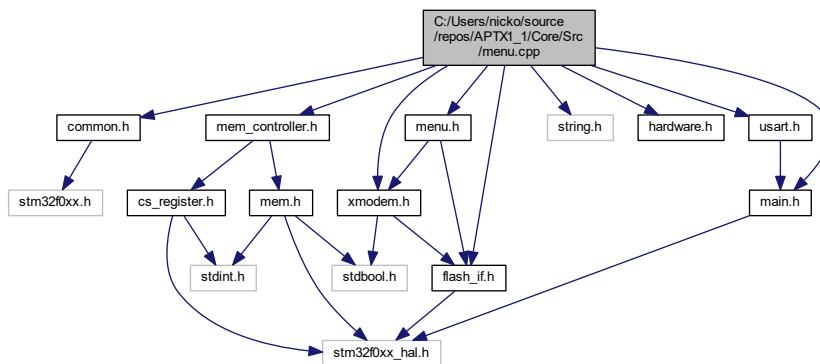
```
#include "uart.h"
Include dependency graph for mem_controller.cpp:
```



6.41 C:/Users/nicko/source/repos/APTX1_1/Core/Src/menu.cpp File Reference

This file provides the software which contains the main menu routine. The main menu gives the options of:

```
#include "main.h"
#include "common.h"
#include "flash_if.h"
#include "menu.h"
#include <xmodem.h>
#include <string.h>
#include "hardware.h"
#include "uart.h"
#include "mem_controller.h"
Include dependency graph for menu.cpp:
```



Functions

- void `dump_mem ()`
- void `Main_Menu (void)`

Display the Main Menu on HyperTerminal.

Variables

- `pFunction JumpToApplication`
- `uint32_t JumpAddress`
- `uint32_t FlashProtection = 0`
- `uint8_t aFileName [FILE_NAME_LENGTH]`
- `Mem_ctrl * memory_extern`

6.41.1 Detailed Description

This file provides the software which contains the main menu routine. The main menu gives the options of:

Author

MCD Application Team

Version

1.0.0

Date

8-April-2015

- downloading a new binary file,
- uploading internal flash memory,
- executing the binary file already loaded
- configuring the write protection of the Flash sectors where the user loads his binary file.

Attention

© COPYRIGHT(c) 2015 STMicroelectronics

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

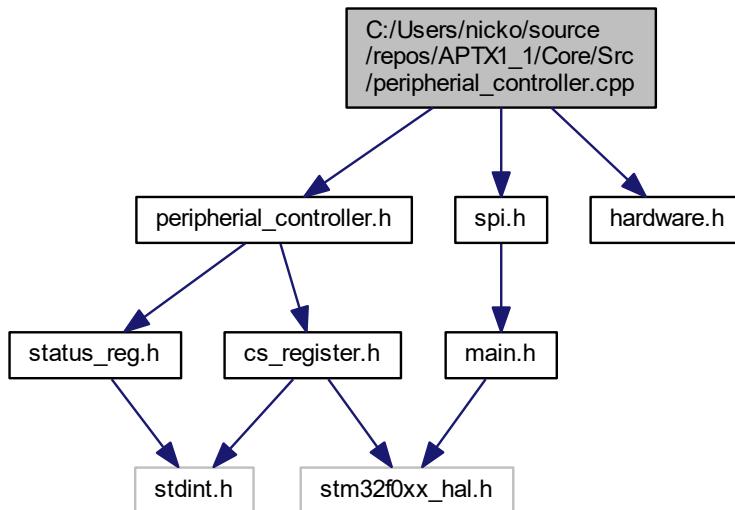
1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.42 C:/Users/nicko/source/repos/APTX1_1/Core/Src/peripheral_controller.cpp File Reference

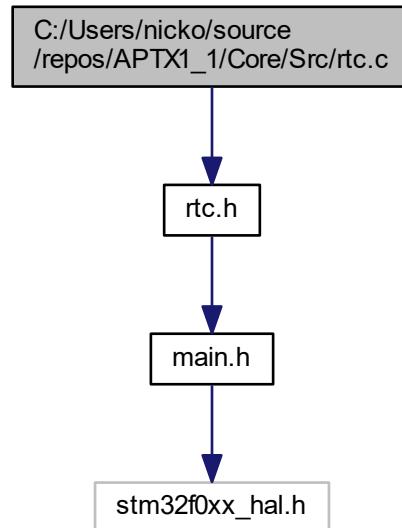
```
#include "peripheral_controller.h"
#include "spi.h"
#include "hardware.h"
```

Include dependency graph for peripheral_controller.cpp:



6.43 C:/Users/nicko/source/repos/APTX1_1/Core/Src/rtc.c File Reference

```
#include "rtc.h"
Include dependency graph for rtc.c:
```



Functions

- void [MX_RTC_Init](#) (void)
- void [HAL_RTC_MspInit](#) (RTC_HandleTypeDef *rtcHandle)
- void [HAL_RTC_MspDeInit](#) (RTC_HandleTypeDef *rtcHandle)

Variables

- RTC_HandleTypeDef [hrtc](#)

6.43.1 Function Documentation

6.43.1.1 HAL_RTC_MspDeInit()

```
void HAL_RTC_MspDeInit (
    RTC_HandleTypeDef * rtcHandle )
```

Definition at line 98 of file rtc.c.

```
99 {
100
101     if(rtcHandle->Instance==RTC)
102     {
103         /* USER CODE BEGIN RTC_MspDeInit 0 */
104
105         /* USER CODE END RTC_MspDeInit 0 */
106         /* Peripheral clock disable */
107         __HAL_RCC_RTC_DISABLE();
108         /* USER CODE BEGIN RTC_MspDeInit 1 */
109
110        /* USER CODE END RTC_MspDeInit 1 */
111    }
112 }
```

6.43.1.2 HAL_RTC_MspInit()

```
void HAL_RTC_MspInit (
    RTC_HandleTypeDef * rtcHandle )
```

Definition at line 82 of file rtc.c.

```
83 {
84
85     if(rtcHandle->Instance==RTC)
86     {
87         /* USER CODE BEGIN RTC_MspInit 0 */
88
89         /* USER CODE END RTC_MspInit 0 */
90         /* RTC clock enable */
91         __HAL_RCC_RTC_ENABLE();
92         /* USER CODE BEGIN RTC_MspInit 1 */
93
94         /* USER CODE END RTC_MspInit 1 */
95     }
96 }
```

6.43.1.3 MX_RTC_Init()

```
void MX_RTC_Init (
    void )
```

Initialize RTC Only

Initialize RTC and set the Time and Date

Enable theTimeStamp

Definition at line 30 of file rtc.c.

```
31 {
32     RTC_TimeTypeDef sTime = {0};
33     RTC_DateTypeDef sDate = {0};
34
35     /** Initialize RTC Only
36     */
37     hrtc.Instance = RTC;
38     hrtc.Init.HourFormat = RTC_HOURFORMAT_24;
39     hrtc.Init.AsynchPrediv = 127;
```

```

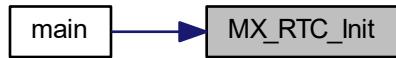
40     hrtc.Init.SynchPrediv = 255;
41     hrtc.Init.OutPut = RTC_OUTPUT_DISABLE;
42     hrtc.Init.OutPutPolarity = RTC_OUTPUT_POLARITY_HIGH;
43     hrtc.Init.OutPutType = RTC_OUTPUT_TYPE_OPENDRAIN;
44     if (HAL_RTC_Init(&hrtc) != HAL_OK)
45     {
46         Error_Handler();
47     }
48
49     /* USER CODE BEGIN Check_RTC_BKUP */
50
51     /* USER CODE END Check_RTC_BKUP */
52
53     /** Initialize RTC and set the Time and Date
54     */
55     sTime.Hours = 0x0;
56     sTime.Minutes = 0x0;
57     sTime.Seconds = 0x0;
58     sTime.DayLightSaving = RTC_DAYLIGHTSAVING_NONE;
59     sTime.StoreOperation = RTC_STOREOPERATION_RESET;
60     if (HAL_RTC_SetTime(&hrtc, &sTime, RTC_FORMAT_BCD) != HAL_OK)
61     {
62         Error_Handler();
63     }
64     sDate.WeekDay = RTC_WEEKDAY_MONDAY;
65     sDate.Month = RTC_MONTH_JANUARY;
66     sDate.Date = 0x1;
67     sDate.Year = 0x0;
68
69     if (HAL_RTC_SetDate(&hrtc, &sDate, RTC_FORMAT_BCD) != HAL_OK)
70     {
71         Error_Handler();
72     }
73     /** Enable theTimeStamp
74     */
75     if (HAL_RTCEx_SetTimeStamp(&hrtc, RTC_TIMESTAMPEDGE_RISING, RTC_TIMESTAMPPIN_DEFAULT) != HAL_OK)
76     {
77         Error_Handler();
78     }
79 }

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.2 Variable Documentation

6.43.2.1 hrtc

```
RTC_HandleTypeDef hrtc
```

File Name : [RTC.c](#) Description : This file provides code for the configuration of the RTC instances.

Attention

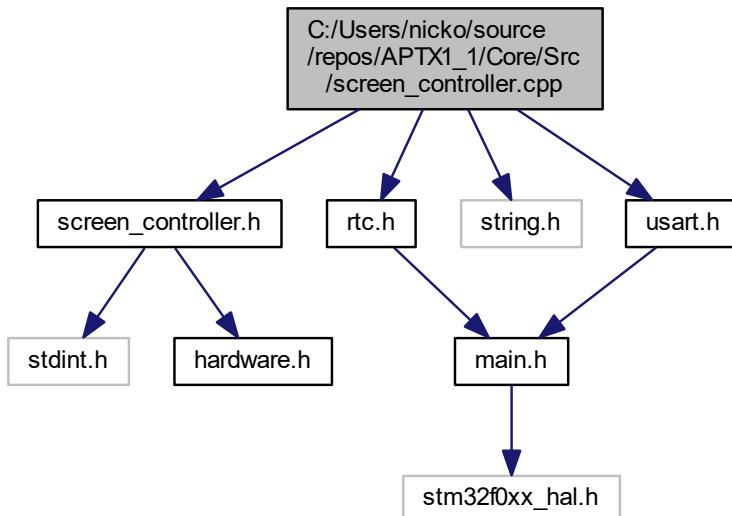
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file rtc.c.

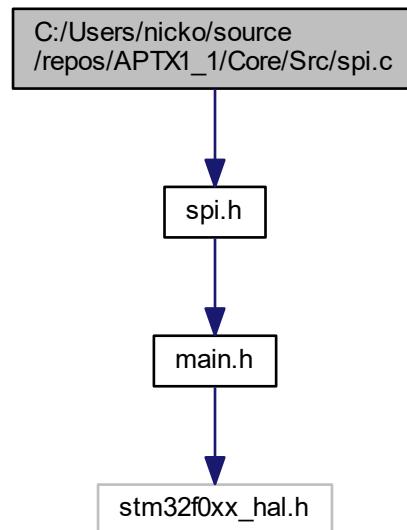
6.44 C:/Users/nicko/source/repos/APTX1_1/Core/Src/screen_controller.cpp File Reference

```
#include "screen_controller.h"
#include "rtc.h"
#include <string.h>
#include "usart.h"
Include dependency graph for screen_controller.cpp:
```



6.45 C:/Users/nicko/source/repos/APTX1_1/Core/Src/spi.c File Reference

```
#include "spi.h"  
Include dependency graph for spi.c:
```



Functions

- void [MX_SPI1_Init](#) (void)
- void [HAL_SPI_MspInit](#) (SPI_HandleTypeDef *spiHandle)
- void [HAL_SPI_MspDeInit](#) (SPI_HandleTypeDef *spiHandle)

Variables

- SPI_HandleTypeDef [hspi1](#)

6.45.1 Function Documentation

6.45.1.1 HAL_SPI_MspDeInit()

```
void HAL_SPI_MspDeInit (
    SPI_HandleTypeDef * spiHandle )

SPI1 GPIO Configuration
PA5 ----> SPI1_SCK PA6 ----> SPI1_MISO PA7 ----> SPI1_MOSI
```

Definition at line 92 of file spi.c.

```
93 {
94
95     if(spiHandle->Instance==SPI1)
96     {
97         /* USER CODE BEGIN SPI1_MspDeInit 0 */
98
99         /* USER CODE END SPI1_MspDeInit 0 */
100        /* Peripheral clock disable */
101        __HAL_RCC_SPI1_CLK_DISABLE();
102
103        /**SPI1 GPIO Configuration
104        PA5      -----> SPI1_SCK
105        PA6      -----> SPI1_MISO
106        PA7      -----> SPI1_MOSI
107        */
108        HAL_GPIO_DeInit(GPIOA, GPIO_PIN_5|GPIO_PIN_6|GPIO_PIN_7);
109
110    /* USER CODE BEGIN SPI1_MspDeInit 1 */
111
112    /* USER CODE END SPI1_MspDeInit 1 */
113 }
114 }
```

6.45.1.2 HAL_SPI_MspInit()

```
void HAL_SPI_MspInit (
    SPI_HandleTypeDef * spiHandle )

SPI1 GPIO Configuration
PA5 ----> SPI1_SCK PA6 ----> SPI1_MISO PA7 ----> SPI1_MOSI
```

Definition at line 54 of file spi.c.

```
55 {
56
57     GPIO_InitTypeDef GPIO_InitStruct = {0};
58     if(spiHandle->Instance==SPI1)
59     {
60         /* USER CODE BEGIN SPI1_MspInit 0 */
61
62         /* USER CODE END SPI1_MspInit 0 */
63         /* SPI1 clock enable */
64         __HAL_RCC_SPI1_CLK_ENABLE();
65
66         __HAL_RCC_GPIOA_CLK_ENABLE();
67         /**SPI1 GPIO Configuration
68         PA5      -----> SPI1_SCK
69         PA6      -----> SPI1_MISO
70         PA7      -----> SPI1_MOSI
71         */
72         GPIO_InitStruct.Pin = GPIO_PIN_5;
73         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
74         GPIO_InitStruct.Pull = GPIO_NOPULL;
75         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
76         GPIO_InitStruct.Alternate = GPIO_AF0_SPI1;
77         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
78
79         GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
80         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
81         GPIO_InitStruct.Pull = GPIO_PULLUP;
82         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
83         GPIO_InitStruct.Alternate = GPIO_AF0_SPI1;
84         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
85
86     /* USER CODE BEGIN SPI1_MspInit 1 */
87
88     /* USER CODE END SPI1_MspInit 1 */
89 }
90 }
```

6.45.1.3 MX_SPI1_Init()

```
void MX_SPI1_Init (
    void )
```

Definition at line 30 of file spi.c.

```
31 {
32
33     hspi1.Instance = SPI1;
34     hspi1.Init.Mode = SPI_MODE_MASTER;
35     hspi1.Init.Direction = SPI_DIRECTION_2LINES;
36     hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
37     hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
38     hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
39     hspi1.Init.NSS = SPI NSS_SOFT;
40     hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_32;
41     hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
42     hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
43     hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
44     hspi1.Init.CRCPolynomial = 7;
45     hspi1.Init.CRCLength = SPI_CRC_LENGTH_DATASIZE;
46     hspi1.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
47     if (HAL_SPI_Init(&hspi1) != HAL_OK)
48     {
49         Error_Handler();
50     }
51 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.45.2 Variable Documentation

6.45.2.1 hspi1

```
SPI_HandleTypeDef hspi1
```

File Name : [SPI.c](#) Description : This file provides code for the configuration of the SPI instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

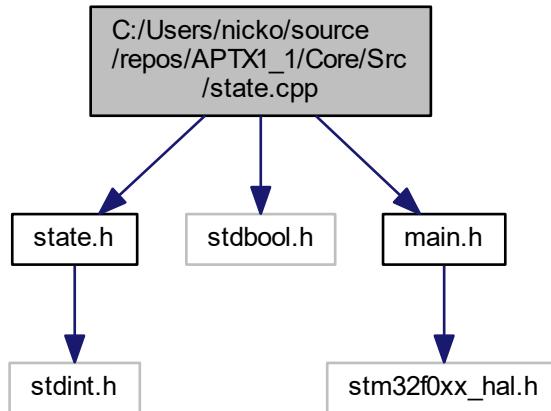
This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file spi.c.

6.46 C:/Users/nicko/source/repos/APTX1_1/Core/Src/state.cpp File Reference

```
#include "state.h"
#include <stdbool.h>
#include "main.h"
```

Include dependency graph for state.cpp:

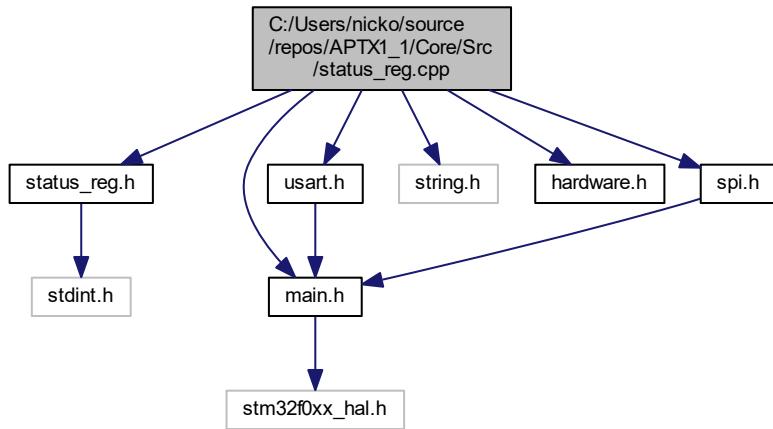


6.47 C:/Users/nicko/source/repos/APTX1_1/Core/Src/status_reg.cpp File Reference

```
#include "status_reg.h"
#include "main.h"
#include "usart.h"
#include <string.h>
#include "hardware.h"
```

```
#include "spi.h"
```

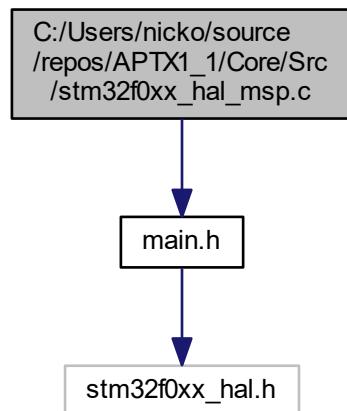
Include dependency graph for status_reg.cpp:



6.48 C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_hal_msp.c File Reference

```
#include "main.h"
```

Include dependency graph for stm32f0xx_hal_msp.c:



Functions

- void HAL_MspInit (void)

6.48.1 Function Documentation

6.48.1.1 HAL_MspInit()

```
void HAL_MspInit (
    void )
```

File Name : [stm32f0xx_hal_msp.c](#) Description : This file provides code for the MSP Initialization and de-Initialization codes.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [opensource.org/licenses/BSD-3-Clause](#) Initializes the Global MSP.

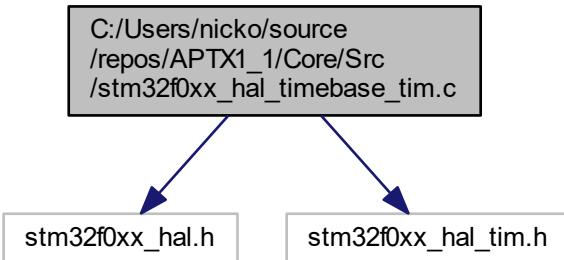
Definition at line 64 of file [stm32f0xx_hal_msp.c](#).

```
65 {
66     /* USER CODE BEGIN MspInit 0 */
67
68     /* USER CODE END MspInit 0 */
69
70     __HAL_RCC_SYSCFG_CLK_ENABLE();
71     __HAL_RCC_PWR_CLK_ENABLE();
72
73     /* System interrupt init*/
74
75     /* USER CODE BEGIN MspInit 1 */
76
77     /* USER CODE END MspInit 1 */
78 }
```

6.49 C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_hal_timebase_tim.c File Reference

HAL time base based on the hardware TIM.

```
#include "stm32f0xx_hal.h"
#include "stm32f0xx_hal_tim.h"
Include dependency graph for stm32f0xx_hal_timebase_tim.c:
```



Functions

- HAL_StatusTypeDef [HAL_InitTick](#) (uint32_t TickPriority)
This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.
- void [HAL_SuspendTick](#) (void)
Suspend Tick increment.
- void [HAL_ResumeTick](#) (void)
Resume Tick increment.

Variables

- TIM_HandleTypeDef [htim1](#)

6.49.1 Detailed Description

HAL time base based on the hardware TIM.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

6.49.2 Function Documentation

6.49.2.1 HAL_InitTick()

```
HAL_StatusTypeDef HAL_InitTick (
    uint32_t TickPriority )
```

This function configures the TIM1 as a time base source. The time source is configured to have 1ms time base with a dedicated Tick interrupt priority.

Note

This function is called automatically at the beginning of program after reset by HAL_Init() or at any time when clock is configured, by HAL_RCC_ClockConfig().

Parameters

<i>TickPriority</i>	Tick interrupt priority.
---------------------	--------------------------

Return values

<i>HAL</i>	status
------------	--------

Definition at line 42 of file stm32f0xx_hal_timebase_tim.c.

```
43 {
44     RCC_ClkInitTypeDef      clkconfig;
45     uint32_t                uwTimclock = 0;
46     uint32_t                uwPrescalerValue = 0;
47     uint32_t                pFLatency;
48
49     /*Configure the TIM1 IRQ priority */
50     HAL_NVIC_SetPriority(TIM1_BRK_UP_TRG_COM_IRQn, TickPriority ,0);
51
52     /* Enable the TIM1 global Interrupt */
53     HAL_NVIC_EnableIRQ(TIM1_BRK_UP_TRG_COM_IRQn);
54
55     /* Enable TIM1 clock */
56     __HAL_RCC_TIM1_CLK_ENABLE();
57
58     /* Get clock configuration */
59     HAL_RCC_GetClockConfig(&clkconfig, &pFLatency);
60
61     /* Compute TIM1 clock */
62     uwTimclock = HAL_RCC_GetPCLK1Freq();
63
64     /* Compute the prescaler value to have TIM1 counter clock equal to 1MHz */
65     uwPrescalerValue = (uint32_t) ((uwTimclock / 1000000) - 1);
66
67     /* Initialize TIM1 */
68     htim1.Instance = TIM1;
69
70     /* Initialize TIMx peripheral as follow:
71     + Period = [(TIM1CLK/1000) - 1]. to have a (1/1000) s time base.
72     + Prescaler = (uwTimclock/1000000 - 1) to have a 1MHz counter clock.
73     + ClockDivision = 0
74     + Counter direction = Up
75     */
76     htim1.Init.Period = (1000000 / 1000) - 1;
77     htim1.Init.Prescaler = uwPrescalerValue;
78     htim1.Init.ClockDivision = 0;
79     htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
80     if(HAL_TIM_Base_Init(&htim1) == HAL_OK)
81     {
82         /* Start the TIM time Base generation in interrupt mode */
```

```

83     return HAL_TIM_Base_Start_IT(&htim1);
84 }
85
86 /* Return function status */
87 return HAL_ERROR;
88 }

```

6.49.2.2 HAL_ResumeTick()

```
void HAL_ResumeTick (
    void )
```

Resume Tick increment.

Note

Enable the tick increment by Enabling TIM1 update interrupt.

Parameters

None	<input type="button" value=""/>
------	---------------------------------

Return values

None	<input type="button" value=""/>
------	---------------------------------

Definition at line 108 of file stm32f0xx_hal_timebase_tim.c.

```

109 {
110     /* Enable TIM1 Update interrupt */
111     __HAL_TIM_ENABLE_IT(&htim1, TIM_IT_UPDATE);
112 }
```

6.49.2.3 HAL_SuspendTick()

```
void HAL_SuspendTick (
    void )
```

Suspend Tick increment.

Note

Disable the tick increment by disabling TIM1 update interrupt.

Parameters

None	<input type="button" value=""/>
------	---------------------------------

Return values

None	
------	--

Definition at line 96 of file stm32f0xx_hal_timebase_tim.c.

```
97 {
98     /* Disable TIM1 update Interrupt */
99     __HAL_TIM_DISABLE_IT(&htim1, TIM_IT_UPDATE);
100 }
```

6.49.3 Variable Documentation

6.49.3.1 htim1

TIM_HandleTypeDef htim1

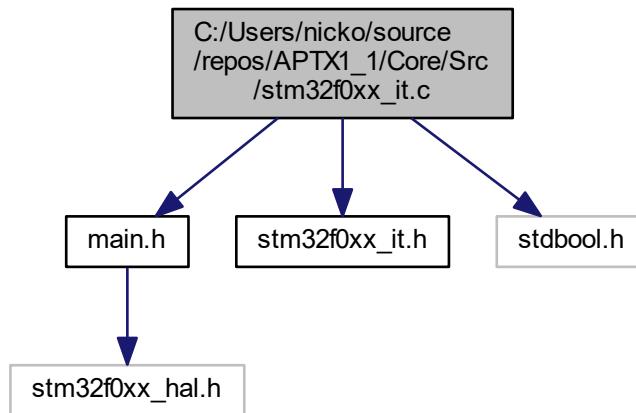
Definition at line 29 of file stm32f0xx_hal_timebase_tim.c.

6.50 C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f0xx_it.h"
#include "stdbool.h"
```

Include dependency graph for stm32f0xx_it.c:



Functions

- `uint8_t read_reset_button (void)`
- `bool test_for_reset_press (void)`
- `void NMI_Handler (void)`

This function handles Non maskable interrupt.
- `void HardFault_Handler (void)`

This function handles Hard fault interrupt.
- `void SVC_Handler (void)`

This function handles System service call via SWI instruction.
- `void PendSV_Handler (void)`

This function handles Pendable request for system service.
- `void SysTick_Handler (void)`

This function handles System tick timer.
- `void DMA1_Channel1_IRQHandler (void)`

This function handles DMA1 channel 1 global interrupt.
- `void TIM1_BRK_UP_TRG_COM_IRQHandler (void)`

This function handles TIM1 break, update, trigger and commutation interrupts.
- `void TIM16_IRQHandler (void)`

This function handles TIM16 global interrupt.
- `void USART1_IRQHandler (void)`

This function handles USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25.

Variables

- `DMA_HandleTypeDef hdma_adc`
- `TIM_HandleTypeDef htim16`
- `UART_HandleTypeDef huart1`
- `TIM_HandleTypeDef htim1`
- `bool reset_pressed`

6.50.1 Detailed Description

Interrupt Service Routines.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

6.50.2 Function Documentation

6.50.2.1 DMA1_Channel1_IRQHandler()

```
void DMA1_Channel1_IRQHandler (
    void )
```

This function handles DMA1 channel 1 global interrupt.

Definition at line 167 of file stm32f0xx_it.c.

```
168 {
169     /* USER CODE BEGIN DMA1_Channel1_IRQHandler_0 */
170
171     /* USER CODE END DMA1_Channel1_IRQHandler_0 */
172     HAL_DMA_IRQHandler(&hdma_adc);
173     /* USER CODE BEGIN DMA1_Channel1_IRQHandler_1 */
174
175     /* USER CODE END DMA1_Channel1_IRQHandler_1 */
176 }
```

6.50.2.2 HardFault_Handler()

```
void HardFault_Handler (
    void )
```

This function handles Hard fault interrupt.

Definition at line 105 of file stm32f0xx_it.c.

```
106 {
107     /* USER CODE BEGIN HardFault_IRQHandler_0 */
108
109     /* USER CODE END HardFault_IRQHandler_0 */
110     while (1)
111     {
112         /* USER CODE BEGIN W1_HardFault_IRQHandler_0 */
113         /* USER CODE END W1_HardFault_IRQHandler_0 */
114     }
115 }
```

6.50.2.3 NMI_Handler()

```
void NMI_Handler (
    void )
```

This function handles Non maskable interrupt.

Definition at line 92 of file stm32f0xx_it.c.

```
93 {
94     /* USER CODE BEGIN NonMaskableInt_IRQHandler_0 */
95
96     /* USER CODE END NonMaskableInt_IRQHandler_0 */
97     /* USER CODE BEGIN NonMaskableInt_IRQHandler_1 */
98
99     /* USER CODE END NonMaskableInt_IRQHandler_1 */
100 }
```

6.50.2.4 PendSV_Handler()

```
void PendSV_Handler (
    void )
```

This function handles Pendable request for system service.

Definition at line 133 of file stm32f0xx_it.c.

```
134 {
135     /* USER CODE BEGIN PendSV_IRQn 0 */
136
137     /* USER CODE END PendSV_IRQn 0 */
138     /* USER CODE BEGIN PendSV_IRQn 1 */
139
140     /* USER CODE END PendSV_IRQn 1 */
141 }
```

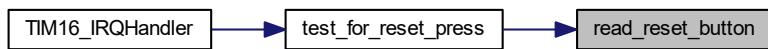
6.50.2.5 read_reset_button()

```
uint8_t read_reset_button (
    void )
```

Definition at line 56 of file stm32f0xx_it.c.

```
57 {
58     return (uint16_t)HAL_GPIO_ReadPin(TOGGLE400_RESET_BTN_GPIO_Port, TOGGLE400_RESET_BTN_Pin);
59 }
```

Here is the caller graph for this function:



6.50.2.6 SVC_Handler()

```
void SVC_Handler (
    void )
```

This function handles System service call via SWI instruction.

Definition at line 120 of file stm32f0xx_it.c.

```
121 {
122     /* USER CODE BEGIN SVC_IRQn 0 */
123
124     /* USER CODE END SVC_IRQn 0 */
125     /* USER CODE BEGIN SVC_IRQn 1 */
126
127     /* USER CODE END SVC_IRQn 1 */
128 }
```

6.50.2.7 SysTick_Handler()

```
void SysTick_Handler (
    void )
```

This function handles System tick timer.

Definition at line 146 of file stm32f0xx_it.c.

```
147 {
148     /* USER CODE BEGIN SysTick_IRQn 0 */
149
150     /* USER CODE END SysTick_IRQn 0 */
151
152     /* USER CODE BEGIN SysTick_IRQn 1 */
153
154     /* USER CODE END SysTick_IRQn 1 */
155 }
```

6.50.2.8 test_for_reset_press()

```
bool test_for_reset_press (
    void )
```

Definition at line 60 of file stm32f0xx_it.c.

```
60
61
62     static uint16_t button_history = 0;
63     bool pressed = false;
64
65     button_history = button_history << 1;
66     button_history |= read_reset_button();
67     if ((button_history & 0xFOFF) == 0xFF)
68     {
69         pressed = true;
70         button_history = 0xFFFF;
71     }
72     return pressed;
73 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.50.2.9 TIM16_IRQHandler()

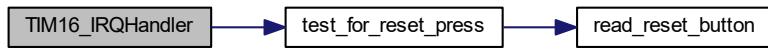
```
void TIM16_IRQHandler (
    void )
```

This function handles TIM16 global interrupt.

Definition at line 195 of file stm32f0xx_it.c.

```
196 {
197     /* USER CODE BEGIN TIM16_IRQHandler_0 */
198
199     if (test_for_reset_press)
200     {
201         reset_pressed = true;
202     }
203
204     /* USER CODE END TIM16_IRQHandler_0 */
205     HAL_TIM_IRQHandler(&htim16);
206     /* USER CODE BEGIN TIM16_IRQHandler_1 */
207
208     /* USER CODE END TIM16_IRQHandler_1 */
209 }
```

Here is the call graph for this function:



6.50.2.10 TIM1_BRK_UP_TRG_COM_IRQHandler()

```
void TIM1_BRK_UP_TRG_COM_IRQHandler (
    void )
```

This function handles TIM1 break, update, trigger and commutation interrupts.

Definition at line 181 of file stm32f0xx_it.c.

```
182 {
183     /* USER CODE BEGIN TIM1_BRK_UP_TRG_COM_IRQHandler_0 */
184
185     /* USER CODE END TIM1_BRK_UP_TRG_COM_IRQHandler_0 */
186     HAL_TIM_IRQHandler(&htim1);
187     /* USER CODE BEGIN TIM1_BRK_UP_TRG_COM_IRQHandler_1 */
188
189     /* USER CODE END TIM1_BRK_UP_TRG_COM_IRQHandler_1 */
190 }
```

6.50.2.11 USART1_IRQHandler()

```
void USART1_IRQHandler (
    void )
```

This function handles USART1 global interrupt / USART1 wake-up interrupt through EXTI line 25.

Definition at line 214 of file stm32f0xx_it.c.

```
215 {
216     /* USER CODE BEGIN USART1_IRQHandler_0 */
217
218     /* USER CODE END USART1_IRQHandler_0 */
219     HAL_UART_IRQHandler(&huart1);
220     /* USER CODE BEGIN USART1_IRQHandler_1 */
221
222     /* USER CODE END USART1_IRQHandler_1 */
223 }
```

6.50.3 Variable Documentation

6.50.3.1 hdma_adc

```
DMA_HandleTypeDef hdma_adc
```

Definition at line 28 of file adc.c.

6.50.3.2 htim1

```
TIM_HandleTypeDef htim1
```

Definition at line 29 of file stm32f0xx_hal_timebase_tim.c.

6.50.3.3 htim16

```
TIM_HandleTypeDef htim16
```

File Name : [TIM.c](#) Description : This file provides code for the configuration of the TIM instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file tim.c.

6.50.3.4 huart1

```
UART_HandleTypeDef huart1
```

File Name : [USART.c](#) Description : This file provides code for the configuration of the USART instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file usart.c.

6.50.3.5 reset_pressed

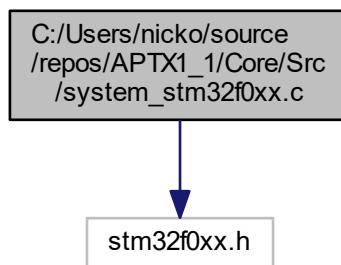
```
bool reset_pressed
```

Definition at line 84 of file main.cpp.

6.51 C:/Users/nicko/source/repos/APTX1_1/Core/Src/system_stm32f0xx.c File Reference

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

```
#include "stm32f0xx.h"  
Include dependency graph for system_stm32f0xx.c:
```



Macros

- #define HSE_VALUE ((uint32_t)8000000)
- #define HSI_VALUE ((uint32_t)8000000)
- #define HSI48_VALUE ((uint32_t)48000000)

Functions

- void [SystemInit](#) (void)
Setup the microcontroller system. Initialize the default HSI clock source, vector table location and the PLL configuration is reset.
- void [SystemCoreClockUpdate](#) (void)
Update SystemCoreClock variable according to Clock Register Values. The SystemCoreClock variable contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.

Variables

- uint32_t [SystemCoreClock](#) = 8000000
- const uint8_t [AHBPrescTable](#) [16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8, 9}
- const uint8_t [APBPrescTable](#) [8] = {0, 0, 0, 0, 1, 2, 3, 4}

6.51.1 Detailed Description

CMSIS Cortex-M0 Device Peripheral Access Layer System Source File.

Author

MCD Application Team

1. This file provides two functions and one global variable to be called from user application:
 - [SystemInit\(\)](#): This function is called at startup just after reset and before branch to main program. This call is made inside the "startup_stm32f0xx.s" file.
 - SystemCoreClock variable: Contains the core clock (HCLK), it can be used by the user application to setup the SysTick timer or configure other parameters.
 - [SystemCoreClockUpdate\(\)](#): Updates the variable SystemCoreClock and must be called whenever the core clock is changed during program execution.
2. After each device reset the HSI (8 MHz) is used as system clock source. Then [SystemInit\(\)](#) function is called, in "startup_stm32f0xx.s" file, to configure the system clock before to branch to main program.

6.51.2 3. This file configures the system clock as follows:

6.51.2.1 Supported STM32F0xx device

6.51.2.2 System Clock source | HSI

6.51.2.3 SYSCLK(Hz) | 8000000

6.51.2.4 HCLK(Hz) | 8000000

6.51.2.5 AHB Prescaler | 1

6.51.2.6 APB1 Prescaler | 1

Attention

© COPYRIGHT(c) 2016 STMicroelectronics

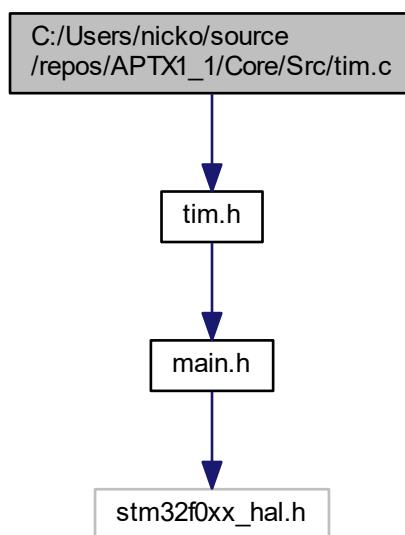
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of STMicroelectronics nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

6.52 C:/Users/nicko/source/repos/APTX1_1/Core/Src/tim.c File Reference

```
#include "tim.h"  
Include dependency graph for tim.c:
```



Functions

- void [MX_TIM16_Init](#) (void)
- void [HAL_TIM_Base_MspInit](#) (TIM_HandleTypeDef *tim_baseHandle)
- void [HAL_TIM_Base_MspDeInit](#) (TIM_HandleTypeDef *tim_baseHandle)

Variables

- TIM_HandleTypeDef [htim16](#)

6.52.1 Function Documentation

6.52.1.1 HAL_TIM_Base_MspDeInit()

```
void HAL_TIM_Base_MspDeInit (
    TIM_HandleTypeDef * tim_baseHandle )
```

Definition at line 95 of file tim.c.

```
96 {
97
98     if(tim_baseHandle->Instance==TIM16)
99     {
100         /* USER CODE BEGIN TIM16_MspDeInit 0 */
101
102         /* USER CODE END TIM16_MspDeInit 0 */
103         /* Peripheral clock disable */
104         __HAL_RCC_TIM16_CLK_DISABLE();
105
106         /* TIM16 interrupt Deinit */
107         HAL_NVIC_DisableIRQ(TIM16_IRQn);
108         /* USER CODE BEGIN TIM16_MspDeInit 1 */
109
110         /* USER CODE END TIM16_MspDeInit 1 */
111     }
112 }
```

6.52.1.2 HAL_TIM_Base_MspInit()

```
void HAL_TIM_Base_MspInit (
    TIM_HandleTypeDef * tim_baseHandle )
```

Definition at line 75 of file tim.c.

```
76 {
77
78     if(tim_baseHandle->Instance==TIM16)
79     {
80         /* USER CODE BEGIN TIM16_MspInit 0 */
81
82         /* USER CODE END TIM16_MspInit 0 */
83         /* TIM16 clock enable */
84         __HAL_RCC_TIM16_CLK_ENABLE();
85
86         /* TIM16 interrupt Init */
87         HAL_NVIC_SetPriority(TIM16_IRQn, 0, 0);
88         HAL_NVIC_EnableIRQ(TIM16_IRQn);
89         /* USER CODE BEGIN TIM16_MspInit 1 */
90
91         /* USER CODE END TIM16_MspInit 1 */
92     }
93 }
```

6.52.1.3 MX_TIM16_Init()

```
void MX_TIM16_Init (
    void )
```

Definition at line 30 of file tim.c.

```
31 {
32     TIM_OC_InitTypeDef sConfigOC = {0};
33     TIM_BreakDeadTimeConfigTypeDef sBreakDeadTimeConfig = {0};
34
35     htim16.Instance = TIM16;
36     htim16.Init.Prescaler = 48000;
37     htim16.Init.CounterMode = TIM_COUNTERMODE_UP;
38     htim16.Init.Period = 10;
39     htim16.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
40     htim16.Init.RepetitionCounter = 5;
41     htim16.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
42     if (HAL_TIM_Base_Init(&htim16) != HAL_OK)
43     {
44         Error_Handler();
45     }
46     if (HAL_TIM_OC_Init(&htim16) != HAL_OK)
47     {
48         Error_Handler();
49     }
50     sConfigOC.OCMode = TIM_OCMODE_TIMING;
51     sConfigOC.Pulse = 0;
52     sConfigOC.OCPPolarity = TIM_OCPOLARITY_HIGH;
53     sConfigOC.OCNPolarity = TIM_OCNPOLARITY_HIGH;
54     sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
55     sConfigOC.OCIdleState = TIM_OCIDLESTATE_RESET;
56     sConfigOC.OCNIdleState = TIM_OCNIDLESTATE_RESET;
57     if (HAL_TIM_OC_ConfigChannel(&htim16, &sConfigOC, TIM_CHANNEL_1) != HAL_OK)
58     {
59         Error_Handler();
60     }
61     sBreakDeadTimeConfig.OffStateRunMode = TIM_OSSR_DISABLE;
62     sBreakDeadTimeConfig.OffStateIDLEMode = TIM_OSSI_DISABLE;
63     sBreakDeadTimeConfig.LockLevel = TIM_LOCKLEVEL_OFF;
64     sBreakDeadTimeConfig.DeadTime = 0;
65     sBreakDeadTimeConfig.BreakState = TIM_BREAK_DISABLE;
66     sBreakDeadTimeConfig.BreakPolarity = TIM_BREAKPOLARITY_HIGH;
67     sBreakDeadTimeConfig.AutomaticOutput = TIM_AUTOMATICOUTPUT_DISABLE;
68     if (HAL_TIMEx_ConfigBreakDeadTime(&htim16, &sBreakDeadTimeConfig) != HAL_OK)
69     {
70         Error_Handler();
71     }
72 }
```

Here is the call graph for this function:



6.52.2 Variable Documentation

6.52.2.1 htim16

```
TIM_HandleTypeDef htim16
```

File Name : [TIM.c](#) Description : This file provides code for the configuration of the TIM instances.

Attention

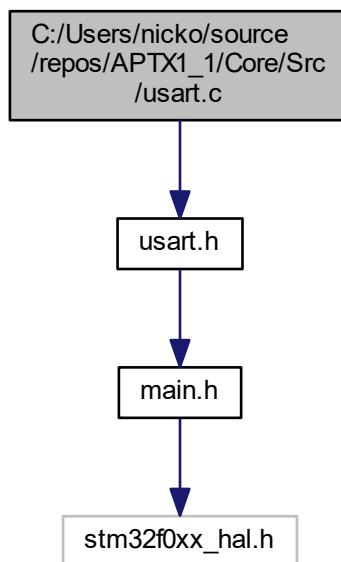
© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file tim.c.

6.53 C:/Users/nicko/source/repos/APTX1_1/Core/Src/usart.c File Reference

```
#include "usart.h"  
Include dependency graph for usart.c:
```



Functions

- void [MX_USART1_UART_Init](#) (void)
- void [MX_USART2_UART_Init](#) (void)
- void [HAL_UART_MspInit](#) (UART_HandleTypeDef *uartHandle)
- void [HAL_UART_MspDeInit](#) (UART_HandleTypeDef *uartHandle)

Variables

- UART_HandleTypeDef [huart1](#)
- UART_HandleTypeDef [huart2](#)

6.53.1 Function Documentation

6.53.1.1 HAL_UART_MspDeInit()

```
void HAL_UART_MspDeInit (
    UART_HandleTypeDef * uartHandle )
```

USART1 GPIO Configuration

PA9 ----> USART1_TX PA10 ----> USART1_RX

USART2 GPIO Configuration

PA2 ----> USART2_TX PA3 ----> USART2_RX

Definition at line 130 of file usart.c.

```
131 {
132
133     if(uartHandle->Instance==USART1)
134     {
135         /* USER CODE BEGIN USART1_MspDeInit 0 */
136
137         /* USER CODE END USART1_MspDeInit 0 */
138         /* Peripheral clock disable */
139         __HAL_RCC_USART1_CLK_DISABLE();
140
141         /**USART1 GPIO Configuration
142          PA9      -----> USART1_TX
143          PA10      -----> USART1_RX
144         */
145         HAL_GPIO_DeInit(GPIOA, UART1\_PC\_TX\_Pin|UART1\_PC\_RX\_Pin);
146
147         /* USART1 interrupt Deinit */
148         HAL_NVIC_DisableIRQ(USART1_IRQn);
149         /* USER CODE BEGIN USART1_MspDeInit 1 */
150
151         /* USER CODE END USART1_MspDeInit 1 */
152     }
153     else if(uartHandle->Instance==USART2)
154     {
155         /* USER CODE BEGIN USART2_MspDeInit 0 */
156
157         /* USER CODE END USART2_MspDeInit 0 */
158         /* Peripheral clock disable */
159         __HAL_RCC_USART2_CLK_DISABLE();
160
161         /**USART2 GPIO Configuration
162          PA2      -----> USART2_TX
163          PA3      -----> USART2_RX
164         */
165         HAL_GPIO_DeInit(GPIOA, UART2\_SCREEN\_TX\_Pin|UART2\_SCREEN\_RX\_Pin);
166
167         /* USER CODE BEGIN USART2_MspDeInit 1 */
168
169         /* USER CODE END USART2_MspDeInit 1 */
170     }
171 }
```

6.53.1.2 HAL_UART_MspInit()

```
void HAL_UART_MspInit (
    UART_HandleTypeDef * uartHandle )
```

USART1 GPIO Configuration
PA9 ----> USART1_TX PA10 ----> USART1_RX

USART2 GPIO Configuration
PA2 ----> USART2_TX PA3 ----> USART2_RX

Definition at line 73 of file `uart.c`.

```
74 {
75
76     GPIO_InitTypeDef GPIO_InitStruct = {0};
77     if(uartHandle->Instance==USART1)
78     {
79         /* USER CODE BEGIN USART1_MspInit 0 */
80
81         /* USER CODE END USART1_MspInit 0 */
82         /* USART1 clock enable */
83         __HAL_RCC_USART1_CLK_ENABLE();
84
85         __HAL_RCC_GPIOA_CLK_ENABLE();
86         /**USART1 GPIO Configuration
87         PA9      -----> USART1_TX
88         PA10      -----> USART1_RX
89         */
90         GPIO_InitStruct.Pin = UART1_PC_TX_Pin|UART1_PC_RX_Pin;
91         GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
92         GPIO_InitStruct.Pull = GPIO_NOPULL;
93         GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
94         GPIO_InitStruct.Alternate = GPIO_AF1_USART1;
95         HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
96
97         /* USART1 interrupt Init */
98         HAL_NVIC_SetPriority(USART1_IRQn, 3, 0);
99         HAL_NVIC_EnableIRQ(USART1_IRQn);
100        /* USER CODE BEGIN USART1_MspInit 1 */
101
102        /* USER CODE END USART1_MspInit 1 */
103    }
104    else if(uartHandle->Instance==USART2)
105    {
106        /* USER CODE BEGIN USART2_MspInit 0 */
107
108        /* USER CODE END USART2_MspInit 0 */
109        /* USART2 clock enable */
110        __HAL_RCC_USART2_CLK_ENABLE();
111
112        __HAL_RCC_GPIOA_CLK_ENABLE();
113        /**USART2 GPIO Configuration
114        PA2      -----> USART2_TX
115        PA3      -----> USART2_RX
116        */
117        GPIO_InitStruct.Pin = UART2_SCREEN_TX_Pin|UART2_SCREEN_RX_Pin;
118        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
119        GPIO_InitStruct.Pull = GPIO_NOPULL;
120        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
121        GPIO_InitStruct.Alternate = GPIO_AF1_USART2;
122        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);
123
124        /* USER CODE BEGIN USART2_MspInit 1 */
125
126        /* USER CODE END USART2_MspInit 1 */
127    }
128 }
```

6.53.1.3 MX_USART1_UART_Init()

```
void MX_USART1_UART_Init (
    void )
```

Definition at line 32 of file usart.c.

```

33 {
34
35     huart1.Instance = USART1;
36     huart1.Init.BaudRate = 115200;
37     huart1.Init.WordLength = UART_WORDLENGTH_8B;
38     huart1.Init.StopBits = UART_STOPBITS_1;
39     huart1.Init.Parity = UART_PARITY_NONE;
40     huart1.Init.Mode = UART_MODE_TX_RX;
41     huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
42     huart1.Init.OverSampling = UART_OVERSAMPLING_16;
43     huart1.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
44     huart1.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
45     if (HAL_UART_Init(&huart1) != HAL_OK)
46     {
47         Error_Handler();
48     }
49 }
50 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.53.1.4 MX_USART2_UART_Init()

```

void MX_USART2_UART_Init (
    void )
```

Definition at line 53 of file usart.c.

```

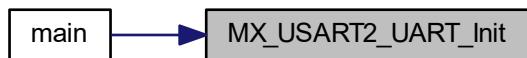
54 {
55
56     huart2.Instance = USART2;
57     huart2.Init.BaudRate = 38400;
58     huart2.Init.WordLength = UART_WORDLENGTH_8B;
59     huart2.Init.StopBits = UART_STOPBITS_1;
60     huart2.Init.Parity = UART_PARITY_NONE;
61     huart2.Init.Mode = UART_MODE_TX_RX;
62     huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
63     huart2.Init.OverSampling = UART_OVERSAMPLING_16;
64     huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
65     huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
```

```
66     if (HAL_UART_Init(&huart2) != HAL_OK)
67     {
68         Error_Handler();
69     }
70 }
71 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.53.2 Variable Documentation

6.53.2.1 huart1

UART_HandleTypeDef huart1

File Name : [USART.c](#) Description : This file provides code for the configuration of the USART instances.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: opensource.org/licenses/BSD-3-Clause

Definition at line 27 of file [usart.c](#).

6.53.2.2 huart2

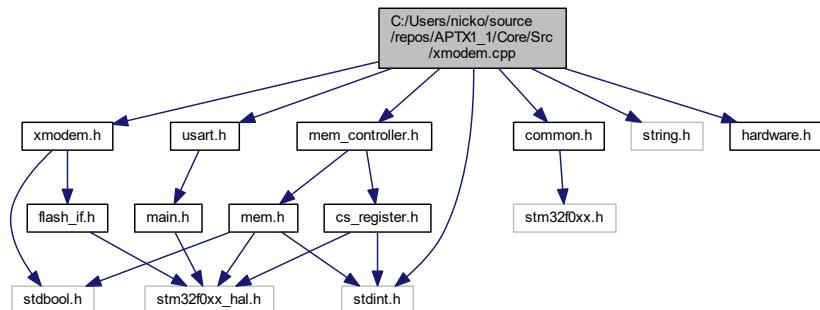
UART_HandleTypeDef huart2

Definition at line 28 of file usart.c.

6.54 C:/Users/nicko/source/repos/APTX1_1/Core/Src/xmodem.cpp File Reference

This module is the implementation of the Xmodem protocol.

```
#include "xmodem.h"
#include <stdint.h>
#include "common.h"
#include <string.h>
#include "hardware.h"
#include "usart.h"
#include "mem_controller.h"
Include dependency graph for xmodem.cpp:
```



Functions

- void [xmodem_receive](#) (void)

This function is the base of the Xmodem protocol. When we receive a header from UART, it decides what action it shall take.
- uint16_t [xmodem_calc_crc](#) (uint8_t *data, uint16_t length)

Calculates the CRC-16 for the input package.

Variables

- uint8_t aPacketData [PACKET_1K_SIZE+PACKET_DATA_INDEX+PACKET_TRAILER_SIZE]
- Mem_ctrl * memory_extern

6.54.1 Detailed Description

This module is the implementation of the Xmodem protocol.

Author

Ferenc Nemeth

Date

21 Dec 2018

Copyright (c) 2018 Ferenc Nemeth - <https://github.com/ferenc-nemeth>

6.54.2 Function Documentation

6.54.2.1 xmodem_calc_crc()

```
uint16_t xmodem_calc_crc (
    uint8_t * data,
    uint16_t length )
```

Calculates the CRC-16 for the input package.

Parameters

<i>*data</i>	Array of the data which we want to calculate.
<i>length</i>	Size of the data, either 128 or 1024 bytes.

Returns

status: The calculated CRC.

Definition at line 155 of file xmodem.cpp.

```
156 {
157     uint16_t crc = 0u;
158     while (length)
159     {
160         length--;
161         crc = crc ^ ((uint16_t)*data++ << 8u);
162         for (uint8_t i = 0u; i < 8u; i++)
163         {
164             if (crc & 0x8000u)
165             {
166                 crc = (crc << 1u) ^ 0x1021u;
167             }
168             else
169             {
170                 crc = crc << 1u;
171             }
172         }
173     }
174     return crc;
175 }
```

6.54.2.2 xmodem_receive()

```
void xmodem_receive (
    void )
```

This function is the base of the Xmodem protocol. When we receive a header from UART, it decides what action it shall take.

Parameters

<code>void</code>	
-------------------	--

Returns

`void`

Definition at line 34 of file xmodem.cpp.

```
35 {
36     volatile xmodem_status status = X_OK;
37     uint8_t error_number = 0u;
38
39     x_first_packet_received = false;
40     xmodem_packet_number = 1u;
41     xmodem_actual_flash_address = APPLICATION_ADDRESS;
42
43     /* Loop until there isn't any error (or until we jump to the user application). */
44     while (X_OK == status)
45     {
46         uint8_t header = 0x00u;
47
48         /* Get the header from UART. */
49         HAL_StatusTypeDef comm_status = HAL_UART_Receive(SERIAL_PC, &header, 1u, 100);
50
51         /* Spam the host (until we receive something) with ACSII "C", to notify it, we want to use
52          CRC-16. */
53         if ((HAL_OK != comm_status) && (false == x_first_packet_received))
54         {
55             (void)Serial_PutByte(X_C);
56
57             /* Uart timeout or any other errors. */
58         else if ((HAL_OK != comm_status) && (true == x_first_packet_received))
59         {
60             status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
61         }
62     else
63     {
64         /* Do nothing. */
65     }
66
67     xmodem_status packet_status = X_ERROR;
68     packet_status = xmodem_handle_packet(header);
69     /* The header can be: SOH, STX, EOT and CAN. */
70
71     switch (header)
72     {
73         /* 128 or 1024 bytes of data. */
74         case X_SOH:
75             /* If the handling was successful, then send an ACK. */
76             if (X_OK == packet_status)
77             {
78                 (void)Serial_PutByte(X_ACK);
79
80             /* If the error was flash related, then immediately set the error counter to max (graceful
81             abort). */
82             else if(X_ERROR_FLASH == packet_status)
83             {
84                 error_number = X_MAX_ERRORS;
85                 status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
86             }
87             /* Error while processing the packet, either send a NAK or do graceful abort. */
88             else
89             {
90                 status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
91             }
92             break;
93         case X_STX:
```

```

92         /* If the handling was successful, then send an ACK. */
93         if (X_OK == packet_status)
94         {
95             (void)Serial_PutByte(X_ACK);
96         }
97         /* If the error was flash related, then immediately set the error counter to max
98        (graceful abort). */
99         else if (X_ERROR_FLASH == packet_status)
100     {
101         error_number = X_MAX_ERRORS;
102         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
103     }
104     /* Error while processing the packet, either send a NAK or do graceful abort. */
105     else
106     {
107         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
108     }
109     break;
110     /* End of Transmission. */
111 case X_EOT:
112     /* ACK, feedback to user (as a text), then jump to user application. */
113     Serial_PutByte(X_ACK);
114     Serial_PutString("\n\rFirmware updated!\n\r");
115
116     // Write out size of app
117     memory_extern->write(1, 0xffff04, xmodem_actual_flash_address);
118     uint8_t size_c[30];
119     sprintf((char*)size_c, "Write Out: %d\r\n", xmodem_actual_flash_address);
120     HAL_UART_Transmit(SERIAL_PC, size_c, strlen((char*)size_c), 10);
121     // Write out version of app
122     uint8_t ver[30];
123     sprintf((char*)ver, "VERSION: %s\r\n", VERSION);
124     HAL_UART_Transmit(SERIAL_PC, ver, strlen((char*)ver), 10);
125
126     memory_extern->write(1, 0xffff08, MCU_VERSION_MAJOR);
127     memory_extern->write(1, 0xffff0c, MCU_VERSION_MINOR);
128     memory_extern->write(1, 0xffff10, MCU_VERSION_PATCH);
129     memory_extern->write(1, 0xffff14, MCU_VERSION_RC);
130
131     Serial_PutString("Jumping to user application...\n\r");
132     HAL_Delay(2500);
133     NVIC_SystemReset();
134     break;
135     /* Abort from host. */
136 case X_CAN:
137     status = X_ERROR;
138     break;
139 default:
140     /* Wrong header. */
141     if (HAL_OK == comm_status)
142     {
143         status = xmodem_error_handler(&error_number, X_MAX_ERRORS);
144     }
145     break;
146 }
147 }
```

Here is the caller graph for this function:



6.54.3 Variable Documentation

6.54.3.1 aPacketData

```
uint8_t aPacketData[PACKET_1K_SIZE+PACKET_DATA_INDEX+PACKET_TRAILER_SIZE]
```

Definition at line 83 of file main.cpp.

Index

__HAL_SYSCFG_REMAPMEMORY_SRAM
 main.cpp, 280
__attribute__
 main.cpp, 280
~AdcController
 AdcController, 52
~CS_reg
 CS_reg, 75
~MainController
 MainController, 82
~Mem_ctrl
 Mem_ctrl, 98
~Memory
 Memory, 113
~Monitor
 Monitor, 124
~PeripheralDevices
 PeripheralDevices, 144
~STATUS_reg
 STATUS_reg, 167
~Screen_ctrl
 Screen_ctrl, 152
~State
 State, 163

ABORT1
 xmodem.h, 251
ABORT2
 xmodem.h, 251
ABS_RETURN
 flash_if.h, 187
ACK
 xmodem.h, 251
Active
 state.h, 227
active_state
 MainController, 82
ADC, 23
 ADC_RESERVED, 23
 ADC_RESERVED2, 23
 FIFOREAD, 24
 HARDWARE_DEFAULT, 24
 SELECT_CHANNEL_0, 24
 SELECT_CHANNEL_1, 24
 SELECT_CHANNEL_2, 24
 SELECT_CHANNEL_3, 24
 SELECT_CHANNEL_4, 25
 SELECT_CHANNEL_5, 25
 SELECT_CHANNEL_6, 25
 SELECT_CHANNEL_7, 25
SELECT_TEST_REFM, 25
SELECT_TEST_REFP, 25
SELECT_TEST_VDIV2, 26
WRITE_CFR, 26
adc
 MainController, 92
adc.c
 hadc, 264
 HAL_ADC_MspDeInit, 261
 HAL_ADC_MspInit, 262
 hdma_adc, 265
 MX_ADC_Init, 263
adc.h
 hadc, 172
 MX_ADC_Init, 170
ADC_Battery
 Globals, 9
adc_command, 47
 b1, 48
 b2, 48
 bits, 48
 bytes, 48
 CLK_SOURCE, 49
 CMR, 49
 CONV_OUTPUT_FORMAT, 49
 CONVERSION_MODE_SELECT, 49
 D11, 49
 EOC_INT_FUNC, 49
 FIFO_TRIG_LEVEL, 50
 INPUT_SELECT_MODE, 50
 SAMPLE_PERIOD, 50
 SWEEP_SEQ_SELECT, 50
 value, 50
adc_controller
 main.cpp, 287
adc_controller.h
 Motor_current_sense, 174
 Pressure, 174
 select_channel, 174
 Temp, 174
 Undefined, 174
 V12, 174
 V5, 174
 V_Motor, 174
 VDD_sense, 174
ADC_CURRENT_SENSE_GPIO_Port
 main.h, 196
ADC_CURRENT_SENSE_Pin
 main.h, 196

ADC_Presure
 Globals, 9

ADC_Presure_20_bar
 Globals, 10

ADC_Presure_27_bar
 Globals, 10

ADC_RESERVED
 ADC, 23

ADC_RESERVED2
 ADC, 23

AdcController, 51
 ~AdcController, 52

AdcController, 52

bat_percentage, 55

battery_volt, 55

channels_buffer, 55

convert_to_volt, 53

cs_register, 55

current_sense, 55

dma_ring_buffer, 56

pressure_check, 53

pressure_ok, 56

read, 54

stall_pressure, 56

temperature_mcu, 56

vbat, 56

vref, 56

address
 Monitor, 141

aFileName
 menu.h, 217

STM32L0xx_IAP, 34

AHBPrescTable
 STM32F0xx_System_Private_Variables, 42

AM_MEM
 Mem_ctrl, 110

am_sn
 MainController, 92

am_valid
 MainController, 93

AM_WARNING_TIMES
 Auxilary, 18

aPacketData
 main.cpp, 287

 xmodem.cpp, 328

APBPrescTable
 STM32F0xx_System_Private_Variables, 42

APP_ADDRESS
 Globals, 10

APP_ADDRESS_P
 Globals, 10

APP_SIZE
 Globals, 10

APPLICATION_ADDRESS
 flash_if.h, 187

apt
 Monitor, 141

APT_MEM

 Mem_ctrl, 110

 apt_sn
 MainController, 93

 apt_valid
 MainController, 93

 Armenta, 8

 assert_param
 stm32f0xx_hal_conf.h, 231

 atp_extern
 main.cpp, 287

 Auxilary, 18
 AM_WARNING_TIMES, 18

 BEEP_COUNTER_0, 18

 BEEP_COUNTER_200, 19

 BEEP_COUNTER_400, 19

 Beep_Every_X_Counting, 19

 Configuration_register_add, 19

 HDC_1080_ADD, 19

 Humidity_register_add, 19

 KEY_PRESS, 20

 MOTOR_1_PWM_OFF, 20

 MOTOR_1_PWM_ON, 20

 MOTOR_1_PWM_ON_50, 20

 MOTOR_1_PWM_ON_75, 20

 RXBUFFERSIZE, 20

 SERIAL_PC, 21

 SERIAL_SCREEN, 21

 START_BEEP, 21

 Temperature_register_add, 21

 TICKS_FOR_UPDATE, 21

 TICKS_SCREEN_UPDATE_LONG, 21

 TICKS_SCREEN_UPDATE_SUPER_LONG, 22

 TIMEOUT_SCREEN, 22

 b1
 adc_command, 48

 channel, 73

 b2
 adc_command, 48

 channel, 73

 bat_percentage
 AdcController, 55

 battery
 MainController, 93

 State, 164

 battery_percent
 MainController, 93

 battery_volt
 AdcController, 55

 BEEP_COUNTER_0
 Auxilary, 18

 BEEP_COUNTER_200
 Auxilary, 19

 BEEP_COUNTER_400
 Auxilary, 19

 Beep_Every_X_Counting
 Auxilary, 19

 binary_and_patch
 Mem_ctrl, 99

bits
 adc_command, 48
 bypass_state, 58
 byte8_t_mem_status, 60
 byte8_t_reg_cs, 65
 byte8_t_reg_readback, 67
 byte8_t_reg_status, 70
BLINKING_LED_GPIO_Port
 main.h, 196
BLINKING_LED_Pin
 main.h, 197
BLOCK_ERASE_32K
 SPI, 14
BLOCK_ERASE_4K
 SPI, 14
BLOCK_ERASE_64K
 SPI, 15
BP0
 byte8_t_mem_status, 60
BP1
 byte8_t_mem_status, 60
BP2
 byte8_t_mem_status, 61
buffer
 Monitor, 142
BUZZER_GPIO_Port
 main.h, 197
BUZZER_Pin
 main.h, 197
bypass
 State, 164
bypass1
 bypass_state, 58
bypass2
 bypass_state, 58
bypass3
 bypass_state, 58
bypass4
 bypass_state, 58
bypass_state, 57
 bits, 58
 bypass1, 58
 bypass2, 58
 bypass3, 58
 bypass4, 58
 reserved, 58
 value, 58
byte8_t_mem_status, 59
 bits, 60
 BP0, 60
 BP1, 60
 BP2, 61
 bytes, 61
 CMP, 61
 LB1, 61
 LB2, 61
 LB3, 61
 QE, 62
RES, 62
RES2, 62
SEC, 62
SRP0, 62
SRP1, 62
status1, 63
status2, 63
TB, 63
value, 63
WEL, 63
WIP, 63
byte8_t_reg_cs, 64
 bits, 65
DECODER, 65
INA_MOTOR, 65
INB_MOTOR, 65
nibble_CS, 65
nibble_PERIPH, 65
nibbles, 66
QC, 66
QG, 66
QH, 66
value, 66
byte8_t_reg_readback, 67
 bits, 67
CS_MEM2, 67
CS_MEM3, 68
CS_STATUS_INV, 68
DECODER, 68
INA_MOTOR, 68
INB_MOTOR, 68
value, 68
byte8_t_reg_status, 69
 bits, 70
FIRE_STATUS, 70
nibble_OPTO, 70
nibble_PERIPH, 70
nibbles, 70
OPTO_A, 70
OPTO_B, 71
OPTO_C, 71
OPTO_D, 71
OPTO_STATUS, 71
RESERVED, 71
SWITCHES_STATUS, 71
value, 72
BYTE_PROGRAM
 SPI, 15
bytes
 adc_command, 48
 byte8_t_mem_status, 61
 channel, 73
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc.h,
 169
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/adc_controller.h,
 173
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/common.h,
 174

C:/Users/nicko/source/repos/APTX1_1/Core/Inc/crc.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/debug_facilities.cpp,
179	271
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/cs_register.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/dma.c,
181	272
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/dma.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/flash_if.cpp,
183	274
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/flash_if.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/gpio.c,
185	276
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/gpio.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/main.cpp,
189	278
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/hardware.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/main_controller.cpp,
192	290
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/mem.cpp,
194	291
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/main_controller.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/mem_controller.cpp,
211	291
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/menu.cpp,
212	292
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/mem_controller.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/peripheral_controller.cpp,
213	294
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/menu.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/rtc.c,
214	295
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/screen_controller.cpp,
217	298
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/monitor.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/spi.c,
218	299
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/peripheral.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/state.cpp,
218	302
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/rtc.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/status_reg.cpp,
219	302
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/screen_controller.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_hal_msp.c,
223	303
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/spi.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_hal_timebase.c,
224	304
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/state.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/stm32f0xx_it.c,
226	308
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/status_reg.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/system_stm32f0xx.c,
228	315
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_hal.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/tim.c,
229	317
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/stm32f0xx_hal.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/usart.c,
238	320
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/tim.h,	C:/Users/nicko/source/repos/APTX1_1/Core/Src/xmodem.cpp,
242	325
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/usart.h,	CA
245	xmodem.h, 251
C:/Users/nicko/source/repos/APTX1_1/Core/Inc/xmodem.h,	channel, 72
248	b1, 73
C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc.c,	b2, 73
260	bytes, 73
C:/Users/nicko/source/repos/APTX1_1/Core/Src/adc_controller.h,	AdcController, 55
265	check_connections
C:/Users/nicko/source/repos/APTX1_1/Core/Src/common.cpp,	Mem_ctrl, 100
266	check_connections_with_printback
C:/Users/nicko/source/repos/APTX1_1/Core/Src/crc.c,	Mem_ctrl, 101
268	check_id
C:/Users/nicko/source/repos/APTX1_1/Core/Src/cs_register.cpp,	
271	

Memory, 113
chip_select
 Monitor, 142
CHIP_SELECT MCU MEM GPIO_Port
 main.h, 197
CHIP_SELECT MCU MEM Pin
 main.h, 197
CHIP_SELECT_RESERVED GPIO_Port
 main.h, 197
CHIP_SELECT_RESERVED_Pin
 main.h, 198
CHIP_SELECT_SERIAL GPIO_Port
 main.h, 198
CHIP_SELECT_SERIAL_Pin
 main.h, 198
clean_screen
 Screen_ctrl, 152
clear_counter
 PeripheralDevices, 144
CLK_SOURCE
 adc_command, 49
CMP
 byte8_t_mem_status, 61
CMR
 adc_command, 49
CMSIS, 36
common.h
 CONVERTDEC, 177
 CONVERTHEX, 177
 CONVERTHEX_ALPHA, 177
 IS_09, 177
 IS_CAP LETTER, 177
 IS_LC LETTER, 177
 ISVALIDDEC, 178
 ISVALIDHEX, 178
 RX_TIMEOUT, 178
 TX_TIMEOUT, 178
Configuration_register_add
 Auxilary, 19
CONV_OUTPUT_FORMAT
 adc_command, 49
CONVERSION_MODE_SELECT
 adc_command, 49
convert_to_volt
 AdcController, 53
CONVERTDEC
 common.h, 177
CONVERTHEX
 common.h, 177
CONVERTHEX_ALPHA
 common.h, 177
CORE, 7
crc.c
 HAL_CRC_MspDelInit, 268
 HAL_CRC_MsplInit, 269
 hcrc, 270
 MX_CRC_Init, 269
crc.h
 hcrc, 180
 MX_CRC_Init, 180
CRC16
 xmodem.h, 251
CS
 Memory, 120
cs1
 CS_reg, 75
cs1_value
 CS_reg, 79
cs2
 CS_reg, 75
cs3
 CS_reg, 76
CS_A2D
 cs_register.h, 182
cs_a2d
 CS_reg, 76
cs_clear_opto_counter
 CS_reg, 77
cs_decoder
 cs_register.h, 182
CS_MEM2
 byte8_t_reg_readback, 67
 cs_register.h, 182
CS_MEM3
 byte8_t_reg_readback, 68
 cs_register.h, 182
CS_READBACK_SER
 cs_register.h, 182
CS_reg, 74
 ~CS_reg, 75
 cs1, 75
 cs1_value, 79
 cs2, 75
 cs3, 76
 cs_a2d, 76
 cs_clear_opto_counter, 77
 CS_reg, 75
 cs_status, 77
 current, 79
 motor_cmd, 78
 print_current, 78
 readback_register, 79
cs_register
 AdcController, 55
 main.cpp, 288
 Mem_ctrl, 110
cs_register.h
 CS_A2D, 182
 cs_decoder, 182
 CS_MEM2, 182
 CS_MEM3, 182
 CS_READBACK_SER, 182
 CS_STATUS, 182
 OPTO_CNT_CLR_N, 182
 Y0, 182
 CS_STATUS

cs_register.h, 182
 cs_status
 CS_reg, 77
 CS_STATUS_INV
 byte8_t_reg_readback, 68
 CsRegister
 PeripheralDevices, 150
 current
 CS_reg, 79
 State, 164
 STATUS_reg, 168
 current_sense
 AdcController, 55
 cursor
 MainController, 93
 Memory, 120
 cycle
 main.cpp, 288

 D11
 adc_command, 49
 DATA_CACHE_ENABLE
 stm32f0xx_hal_conf.h, 231
 date
 Memory, 120
 debug
 main.cpp, 288
 MainController, 94
 DEBUG_AM_OK_GPIO_Port
 main.h, 198
 DEBUG_AM_OK_Pin
 main.h, 198
 DEBUG_APT_OK_GPIO_Port
 main.h, 198
 DEBUG_APT_OK_Pin
 main.h, 199
 DEBUG_BRK1_GPIO_Port
 main.h, 199
 DEBUG_BRK1_Pin
 main.h, 199
 DEBUG_BRK2_GPIO_Port
 main.h, 199
 DEBUG_BRK2_Pin
 main.h, 199
 DEBUG_CLK_AUX_GPIO_Port
 main.h, 199
 DEBUG_CLK_AUX_Pin
 main.h, 200
 DEBUG_DATA_AUX_GPIO_Port
 main.h, 200
 DEBUG_DATA_AUX_Pin
 main.h, 200
 DEBUG_LATCH_AUX_GPIO_Port
 main.h, 200
 DEBUG_LATCH_AUX_Pin
 main.h, 200
 DEBUG MCU_OK_GPIO_Port
 main.h, 200
 DEBUG MCU_OK_Pin

 main.h, 201
 DEBUG_MOTOR_RUNNING_GPIO_Port
 main.h, 201
 DEBUG_MOTOR_RUNNING_Pin
 main.h, 201
 DEBUG_PRESSURE_OK_GPIO_Port
 main.h, 201
 DEBUG_PRESSURE_OK_Pin
 main.h, 201
 DEBUG_PRINT
 Globals, 11
 DEBUG_STATE_1_GPIO_Port
 main.h, 201
 DEBUG_STATE_1_Pin
 main.h, 202
 DEBUG_STATE_2_GPIO_Port
 main.h, 202
 DEBUG_STATE_2_Pin
 main.h, 202
 DEBUG_STATE_3_GPIO_Port
 main.h, 202
 DEBUG_STATE_3_Pin
 main.h, 202
 DECODER
 byte8_t_reg_cs, 65
 byte8_t_reg_readback, 68
 device
 MainController, 94
 dma.c
 MX_DMA_Init, 273
 dma.h
 MX_DMA_Init, 183
 DMA1_Channel1_IRQHandler
 stm32f0xx_it.c, 310
 stm32f0xx_it.h, 239
 DMA_LENGTH
 main.h, 202
 dma_ring_buffer
 AdcController, 56
 DONT_BEEP
 Globals, 11
 DOWNLOAD_TIMEOUT
 xmodem.h, 251
 dump_mem
 STM32L0xx_IAP, 31

 ENABLE_POWER_12V_GPIO_Port
 main.h, 203
 ENABLE_POWER_12V_Pin
 main.h, 203
 END_ADDRESS
 Globals, 11
 EOC_INT_FUNC
 adc_command, 49
 EOT
 xmodem.h, 252
 erase
 Mem_ctrl, 101, 102
 Memory, 113, 114

adc.c, 262
HAL_CORTEX_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 232
HAL_CRC_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 232
HAL_CRC_MspDeInit
 crc.c, 268
HAL_CRC_MspInit
 crc.c, 269
HAL_DMA_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 232
HAL_FLASH_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 232
HAL_GPIO_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 232
HAL_I2C_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 233
HAL_InitTick
 stm32f0xx_hal_timebase_tim.c, 306
HAL_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 233
HAL_MspInit
 stm32f0xx_hal_msp.c, 304
HAL_PWR_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 233
HAL_RCC_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 233
HAL_ResumeTick
 stm32f0xx_hal_timebase_tim.c, 307
HAL_RTC_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 233
HAL_RTC_MspDeInit
 rtc.c, 295
HAL_RTC_MspInit
 rtc.c, 296
HAL_SPI_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 233
HAL_SPI_MspDeInit
 spi.c, 299
HAL_SPI_MspInit
 spi.c, 300
HAL_SuspendTick
 stm32f0xx_hal_timebase_tim.c, 307
HAL_TIM_Base_MspDeInit
 tim.c, 318
HAL_TIM_Base_MspInit
 tim.c, 318
HAL_TIM_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 234
HAL_TIM_PeriodElapsedCallback
 main.cpp, 281
HAL_UART_MODULE_ENABLED
 stm32f0xx_hal_conf.h, 234
HAL_UART_MspDeInit
 usart.c, 321
HAL_UART_MspInit
 usart.c, 321
HardFault_Handler
 stm32f0xx_it.c, 310
 stm32f0xx_it.h, 239
HARDWARE_DEFAULT
 ADC, 24
hcrc
 crc.c, 270
 crc.h, 180
HDC_1080_ADD
 Auxilary, 19
hdma_adc
 adc.c, 265
 stm32f0xx_it.c, 314
HIGH
 SPI, 15
hrtc
 rtc.c, 297
 rtc.h, 222
HSE_STARTUP_TIMEOUT
 stm32f0xx_hal_conf.h, 234
HSE_VALUE
 stm32f0xx_hal_conf.h, 234
 STM32F0xx_System_Private_Defines, 40
HSI14_VALUE
 stm32f0xx_hal_conf.h, 234
HSI48_VALUE
 stm32f0xx_hal_conf.h, 235
 STM32F0xx_System_Private_Defines, 40
HSI_STARTUP_TIMEOUT
 stm32f0xx_hal_conf.h, 235
HSI_VALUE
 stm32f0xx_hal_conf.h, 235
 STM32F0xx_System_Private_Defines, 40
hspi1
 spi.c, 301
 spi.h, 225
htim1
 stm32f0xx_hal_timebase_tim.c, 308
 stm32f0xx_it.c, 314
htim16
 stm32f0xx_it.c, 314
 tim.c, 319
 tim.h, 244
huart1
 stm32f0xx_it.c, 314
 usart.c, 324
 usart.h, 247
huart2
 usart.c, 324
 usart.h, 248
Humidity_register_add
 Auxilary, 19
id
 Memory, 120
id_valid
 Memory, 121
Idle
 state.h, 227
idle_state

MainController, 84
INA_MOTOR
 byte8_t_reg_cs, 65
 byte8_t_reg_readback, 68
INB_MOTOR
 byte8_t_reg_cs, 65
 byte8_t_reg_readback, 68
Init
 state.h, 227
init_before_while
 MainController, 85
init_first_step
 State, 163
init_state
 MainController, 86
INPUT_SELECT_MODE
 adc_command, 50
INSTRUCTION_CACHE_ENABLE
 stm32f0xx_hal_conf.h, 235
Int2Str
 STM32L0xx_IAP_Main, 27
IS_09
 common.h, 177
IS_CAP_LETTER
 common.h, 177
is_counting_up
 MainController, 94
IS_LC_LETTER
 common.h, 177
ISVALIDDEC
 common.h, 178
ISVALIDHEX
 common.h, 178

JumpAddress
 STM32L0xx_IAP, 35
JumpToApplication
 STM32L0xx_IAP, 35

KEY_PRESS
 Auxilary, 20

LB1
 byte8_t_mem_status, 61
LB2
 byte8_t_mem_status, 61
LB3
 byte8_t_mem_status, 61
LOGIC_BYPASS_1_GPIO_Port
 main.h, 203
LOGIC_BYPASS_1_Pin
 main.h, 203
LOGIC_BYPASS_2_GPIO_Port
 main.h, 203
LOGIC_BYPASS_2_Pin
 main.h, 203
LOGIC_BYPASS_3_GPIO_Port
 main.h, 204
LOGIC_BYPASS_3_Pin
 main.h, 204
LOGIC_BYPASS_4_GPIO_Port
 main.h, 204
LOGIC_BYPASS_4_Pin
 main.h, 204
LOW
 SPI, 15
LSE_STARTUP_TIMEOUT
 stm32f0xx_hal_conf.h, 236
LSE_VALUE
 stm32f0xx_hal_conf.h, 236
LSI_VALUE
 stm32f0xx_hal_conf.h, 236
LVDS_RESERVED_GPIO_Port
 main.h, 204
LVDS_RESERVED_Pin
 main.h, 204

main
 main.cpp, 282
main.cpp
 __HAL_SYSCFG_REMAPMEMORY_SRAM, 280
 __attribute__, 280
 adc_controller, 287
 aPacketData, 287
 atp_extern, 287
 cs_register, 288
 cycle, 288
 debug, 288
 Error_Handler, 280
 grand_ctrl, 288
 HAL_ADC_ConvCpltCallback, 281
 HAL_TIM_PeriodElapsedCallback, 281
 main, 282
 memory_am, 288
 memory_apt, 288
 memory_controller, 289
 memory_mcu, 289
 ph_device, 289
 remapMemToSRAM, 285
 reset_pressed, 289
 screen, 289
 state, 289
 status_register, 290
 SystemClock_Config, 286
 timer_last_update, 290
main.h
 ADC_CURRENT_SENSE_GPIO_Port, 196
 ADC_CURRENT_SENSE_Pin, 196
 BLINKING_LED_GPIO_Port, 196
 BLINKING_LED_Pin, 197
 BUZZER_GPIO_Port, 197
 BUZZER_Pin, 197
 CHIP_SELECT MCU MEM GPIO_Port, 197
 CHIP_SELECT MCU MEM Pin, 197
 CHIP_SELECT RESERVED GPIO_Port, 197
 CHIP_SELECT RESERVED Pin, 198
 CHIP_SELECT SERIAL GPIO_Port, 198
 CHIP_SELECT SERIAL Pin, 198

DEBUG_AM_OK_GPIO_Port, 198
 DEBUG_AM_OK_Pin, 198
 DEBUG_APT_OK_GPIO_Port, 198
 DEBUG_APT_OK_Pin, 199
 DEBUG_BRK1_GPIO_Port, 199
 DEBUG_BRK1_Pin, 199
 DEBUG_BRK2_GPIO_Port, 199
 DEBUG_BRK2_Pin, 199
 DEBUG_CLK_AUX_GPIO_Port, 199
 DEBUG_CLK_AUX_Pin, 200
 DEBUG_DATA_AUX_GPIO_Port, 200
 DEBUG_DATA_AUX_Pin, 200
 DEBUG_LATCH_AUX_GPIO_Port, 200
 DEBUG_LATCH_AUX_Pin, 200
 DEBUG MCU_OK_GPIO_Port, 200
 DEBUG MCU_OK_Pin, 201
 DEBUG_MOTOR_RUNNING_GPIO_Port, 201
 DEBUG_MOTOR_RUNNING_Pin, 201
 DEBUG_PRESSURE_OK_GPIO_Port, 201
 DEBUG_PRESSURE_OK_Pin, 201
 DEBUG_STATE_1_GPIO_Port, 201
 DEBUG_STATE_1_Pin, 202
 DEBUG_STATE_2_GPIO_Port, 202
 DEBUG_STATE_2_Pin, 202
 DEBUG_STATE_3_GPIO_Port, 202
 DEBUG_STATE_3_Pin, 202
 DMA_LENGTH, 202
 ENABLE_POWER_12V_GPIO_Port, 203
 ENABLE_POWER_12V_Pin, 203
 Error_Handler, 210
 LOGIC_BYPASS_1_GPIO_Port, 203
 LOGIC_BYPASS_1_Pin, 203
 LOGIC_BYPASS_2_GPIO_Port, 203
 LOGIC_BYPASS_2_Pin, 203
 LOGIC_BYPASS_3_GPIO_Port, 204
 LOGIC_BYPASS_3_Pin, 204
 LOGIC_BYPASS_4_GPIO_Port, 204
 LOGIC_BYPASS_4_Pin, 204
 LVDS_RESERVED_GPIO_Port, 204
 LVDS_RESERVED_Pin, 204
 PA1_RESERVED_ASK_SHAUL_GPIO_Port, 205
 PA1_RESERVED_ASK_SHAUL_Pin, 205
 POWER_FAIL MCU_GPIO_Port, 205
 POWER_FAIL MCU_Pin, 205
 RESERVED_UART6_RX_GPIO_Port, 205
 RESERVED_UART6_RX_Pin, 205
 RESERVED_UART6_TX_GPIO_Port, 206
 RESERVED_UART6_TX_Pin, 206
 RTC_TIMESTAMP_GPIO_Port, 206
 RTC_TIMESTAMP_Pin, 206
 SIMPLE_INTERLOCK_GPIO_Port, 206
 SIMPLE_INTERLOCK_Pin, 206
 SYS_SWO_RESERVED_GPIO_Port, 207
 SYS_SWO_RESERVED_Pin, 207
 TOGGLE400_RESET_BTN_GPIO_Port, 207
 TOGGLE400_RESET_BTN_Pin, 207
 UART1_PC_RX_GPIO_Port, 207
 UART1_PC_RX_Pin, 207
 UART1_PC_TX_GPIO_Port, 208
 UART1_PC_TX_Pin, 208
 UART2_SCREEN_RX_GPIO_Port, 208
 UART2_SCREEN_RX_Pin, 208
 UART2_SCREEN_TX_GPIO_Port, 208
 UART2_SCREEN_TX_Pin, 208
 USB_DM_RESERVED_GPIO_Port, 209
 USB_DM_RESERVED_Pin, 209
 USB_DP_RESERVED_GPIO_Port, 209
 USB_DP_RESERVED_Pin, 209
 WATCHDOG_OUT_GPIO_Port, 209
 WATCHDOG_OUT_Pin, 209
 main_controller.cpp
 reset_pressed, 291
 Main_Menu
 STM32L0xx_IAP, 33
 MainController, 80
 ~MainController, 82
 active_state, 82
 adc, 92
 am_sn, 92
 am_valid, 93
 apt_sn, 93
 apt_valid, 93
 battery, 93
 battery_percent, 93
 cursor, 93
 debug, 94
 device, 94
 idle_state, 84
 init_before_while, 85
 init_state, 86
 is_counting_up, 94
 MainController, 81
 max_count, 94
 mcu_sn, 94
 mcu_valid, 94
 mem, 95
 motor_running, 95
 need_to_stall, 88
 pressure, 95
 pressure_ok, 95
 pulses, 95
 pulses_on_screen, 95
 qa_state, 88
 screen, 96
 state, 96
 status, 96
 uart_state, 90
 update_state, 91
 max_count
 MainController, 94
 Memory, 121
 MAX_ERRORS
 xmodem.h, 252
 MAX_HOLE_SIZE
 Globals, 12
 MCU_MEM

Mem_ctrl, 110
mcu_sn
 MainController, 94
mcu_valid
 MainController, 94
MCU_VERSION_MAJOR
 Globals, 12
MCU_VERSION_MINOR
 Globals, 12
MCU_VERSION_PATCH
 Globals, 12
MCU_VERSION_RC
 Globals, 12
mem
 MainController, 95
Mem_ctrl, 97
 ~Mem_ctrl, 98
 AM_MEM, 110
 APT_MEM, 110
 binary_and_patch, 99
 check_connections, 100
 check_connections_with_printback, 101
 cs_register, 110
 erase, 101, 102
 get_date, 103
 get_max, 103
 get_serial, 103
 MCU_MEM, 110
 Mem_ctrl, 98
 poll_complete, 104
 print_register, 105
 read, 106
 read256, 107
 write, 107
 write_register, 109
Memory, 111
 ~Memory, 113
 check_id, 113
 CS, 120
 cursor, 120
 date, 120
 erase, 113, 114
 id, 120
 id_valid, 121
 max_count, 121
 Memory, 112
 name, 121
 print_status, 114
 pulses, 121
 read, 115
 read256, 116
 read_status_register1, 116
 read_status_register2, 117
 serial_number, 121
 status16_t, 121
 write, 118
 write_disable, 118
 write_enable, 119
 write_status_register, 119
memory_am
 main.cpp, 288
memory_apt
 main.cpp, 288
memory_controller
 main.cpp, 289
memory_extern
 STM32L0xx_IAP, 35
memory_mcu
 main.cpp, 289
menu.h
 aFileName, 217
 pFunction, 217
 SerialDownload, 217
Monitor, 122
 ~Monitor, 124
 address, 141
 apt, 141
 buffer, 142
 chip_select, 142
 Monitor, 124
 rx_debug, 125
 rx_erase, 125
 rx_exit, 126
 rx_find_pulses, 126
 rx_get_man_id, 127
 rx_go_active, 127
 rx_help, 128
 rx_nuke, 128
 rx_pass, 129
 rx_read, 130
 rx_scan, 130, 131
 rx_set_date, 131
 rx_set_maxi, 132
 rx_set_serial, 133
 rx_start_motor, 134
 rx_status_read, 134
 rx_status_write, 135
 rx_stop_motor, 136
 rx_test_read, 136
 rx_upload, 137
 rx_write, 138
 serial_consume, 138
 value, 142
MOTOR_1_PWM_OFF
 Auxilary, 20
MOTOR_1_PWM_ON
 Auxilary, 20
MOTOR_1_PWM_ON_50
 Auxilary, 20
MOTOR_1_PWM_ON_75
 Auxilary, 20
motor_cmd
 CS_reg, 78
Motor_current_sense
 adc_controller.h, 174
motor_running

MainController, 95
 PeripheralDevices, 150
MX_ADC_Init
 adc.c, 263
 adc.h, 170
MX_CRC_Init
 crc.c, 269
 crc.h, 180
MX_DMA_Init
 dma.c, 273
 dma.h, 183
MX_GPIO_Init
 gpio.c, 276
 gpio.h, 189
MX_RTC_Init
 rtc.c, 296
 rtc.h, 220
MX_SPI1_Init
 spi.c, 300
 spi.h, 224
MX_TIM16_Init
 tim.c, 318
 tim.h, 243
MX_USART1_UART_Init
 usart.c, 322
 usart.h, 246
MX_USART2_UART_Init
 usart.c, 323
 usart.h, 246

NAK
 xmodem.h, 252
NAK_TIMEOUT
 xmodem.h, 252
name
 Memory, 121
need_to_stall
 MainController, 88
NEGATIVE_BYTE
 xmodem.h, 253
next
 State, 164
nibble_CS
 byte8_t_reg_cs, 65
nibble_OPTO
 byte8_t_reg_status, 70
nibble_PERIPH
 byte8_t_reg_cs, 65
 byte8_t_reg_status, 70
nibbles
 byte8_t_reg_cs, 66
 byte8_t_reg_status, 70
NMI_Handler
 stm32f0xx_it.c, 310
 stm32f0xx_it.h, 239

OPTO_A
 byte8_t_reg_status, 70
OPTO_B

byte8_t_reg_status, 71
OPTO_C
 byte8_t_reg_status, 71
OPTO_CNT_CLR_N
 cs_register.h, 182
OPTO_D
 byte8_t_reg_status, 71
OPTO_STATUS
 byte8_t_reg_status, 71

PA1_RESERVED_ASK_SHAUL_GPIO_Port
 main.h, 205
PA1_RESERVED_ASK_SHAUL_Pin
 main.h, 205
PACKET_1K_SIZE
 xmodem.h, 253
PACKET_CNUMBER_INDEX
 xmodem.h, 253
PACKET_DATA_INDEX
 xmodem.h, 253
PACKET_HEADER_SIZE
 xmodem.h, 253
PACKET_NUMBER_INDEX
 xmodem.h, 253
PACKET_OVERHEAD_SIZE
 xmodem.h, 254
PACKET_SIZE
 xmodem.h, 254
PACKET_START_INDEX
 xmodem.h, 254
PACKET_TRAILER_SIZE
 xmodem.h, 254
PASS_PRESSURE_CHECK
 Globals, 13
PASS_SPI
 Globals, 13
pass_to_ard
 Screen_ctrl, 153
PendSV_Handler
 stm32f0xx_it.c, 310
 stm32f0xx_it.h, 240

PeripheralDevices, 143
 ~PeripheralDevices, 144
 clear_counter, 144
 CsRegister, 150
 motor_running, 150
 PeripheralDevices, 144
 read, 145
 read_counter, 146
 read_fire_button_press, 147
 start_motor, 147
 StatusReg, 150
 stop_motor, 148
 toggle_motor, 149

pFunction
 menu.h, 217
ph_device
 main.cpp, 289
poll_complete

Mem_ctrl, 104
POWER_FAIL MCU GPIO_Port
 main.h, 205
POWER_FAIL MCU Pin
 main.h, 205
PREFETCH_ENABLE
 stm32f0xx_hal_conf.h, 236
Pressure
 adc_controller.h, 174
pressure
 MainController, 95
 State, 165
pressure_check
 AdcController, 53
pressure_ok
 AdcController, 56
 MainController, 95
print_current
 CS_reg, 78
 STATUS_reg, 167
print_logo
 Screen_ctrl, 153
print_register
 Mem_ctrl, 105
print_status
 Memory, 114
print_version
 Screen_ctrl, 154
pulses
 MainController, 95
 Memory, 121
pulses_on_screen
 MainController, 95
pulses_to_screen
 State, 165

QA
 state.h, 227
qa_mon
 Screen_ctrl, 154
qa_state
 MainController, 88
QC
 byte8_t_reg_cs, 66
QE
 byte8_t_mem_status, 62
QG
 byte8_t_reg_cs, 66
QH
 byte8_t_reg_cs, 66

read
 AdcController, 54
 Mem_ctrl, 106
 Memory, 115
 PeripheralDevices, 145
read256
 Mem_ctrl, 107
 Memory, 116

 READ_ARRAY1
 SPI, 16
 READ_ARRAY2
 SPI, 16
 read_counter
 PeripheralDevices, 146
READ_DEVICE_ID
 SPI, 16
READ_DUAL
 SPI, 16
read_fire_button_press
 PeripheralDevices, 147
READ_MANUFACTURE_ID
 SPI, 16
READ_QUAD
 SPI, 16
read_reset_button
 stm32f0xx_it.c, 311
READ_STATUS_REGISTER1
 SPI, 17
read_status_register1
 Memory, 116
READ_STATUS_REGISTER2
 SPI, 17
read_status_register2
 Memory, 117
readback_register
 CS_reg, 79
remapMemToSRAM
 main.cpp, 285
RES
 byte8_t_mem_status, 62
RES2
 byte8_t_mem_status, 62
RESERVED
 byte8_t_reg_status, 71
reserved
 bypass_state, 58
RESERVED_UART6_RX_GPIO_Port
 main.h, 205
RESERVED_UART6_RX_Pin
 main.h, 205
RESERVED_UART6_TX_GPIO_Port
 main.h, 206
RESERVED_UART6_TX_Pin
 main.h, 206
reset_pressed
 main.cpp, 289
 main_controller.cpp, 291
 stm32f0xx_it.c, 315
rtc.c
 HAL_RTC_MspDeInit, 295
 HAL_RTC_MspInit, 296
 hrtc, 297
 MX_RTC_Init, 296
rtc.h
 hrtc, 222
 MX_RTC_Init, 220

RTC_TIMESTAMP_GPIO_Port
 main.h, 206
RTC_TIMESTAMP_Pin
 main.h, 206
rx_debug
 Monitor, 125
rx_erase
 Monitor, 125
rx_exit
 Monitor, 126
rx_find_pulses
 Monitor, 126
rx_get_man_id
 Monitor, 127
rx_go_active
 Monitor, 127
rx_help
 Monitor, 128
rx_nuke
 Monitor, 128
rx_pass
 Monitor, 129
rx_read
 Monitor, 130
rx_scan
 Monitor, 130, 131
rx_set_date
 Monitor, 131
rx_set_maxi
 Monitor, 132
rx_set_serial
 Monitor, 133
rx_start_motor
 Monitor, 134
rx_status_read
 Monitor, 134
rx_status_write
 Monitor, 135
rx_stop_motor
 Monitor, 136
rx_test_read
 Monitor, 136
RX_TIMEOUT
 common.h, 178
rx_upload
 Monitor, 137
rx_write
 Monitor, 138
RXBUFFERSIZE
 Auxilary, 20
SAMPLE_PERIOD
 adc_command, 50
screen
 main.cpp, 289
 MainController, 96
Screen_ctrl, 151
 ~Screen_ctrl, 152
 clean_screen, 152
get_time, 153
pass_to_ard, 153
print_logo, 153
print_version, 154
qa_mon, 154
Screen_ctrl, 152
tx_send_general_error, 155
update_battery, 156
update_error_remaining, 157
update_piazo, 158
update_pressure, 159
update_pulse_counter, 160
update_remaining_in_percent, 160
SEC
 byte8_t_mem_status, 62
select_channel
 adc_controller.h, 174
SELECT_CHANNEL_0
 ADC, 24
SELECT_CHANNEL_1
 ADC, 24
SELECT_CHANNEL_2
 ADC, 24
SELECT_CHANNEL_3
 ADC, 24
SELECT_CHANNEL_4
 ADC, 25
SELECT_CHANNEL_5
 ADC, 25
SELECT_CHANNEL_6
 ADC, 25
SELECT_CHANNEL_7
 ADC, 25
SELECT_TEST_REFM
 ADC, 25
SELECT_TEST_REFP
 ADC, 25
SELECT_TEST_VDIV2
 ADC, 26
serial_consume
 Monitor, 138
serial_number
 Memory, 121
SERIAL_PC
 Auxilary, 21
Serial_PutByte
 STM32L0xx_IAP_Main, 28
Serial_PutString
 STM32L0xx_IAP_Main, 28
SERIAL_SCREEN
 Auxilary, 21
SerialDownload
 menu.h, 217
SIMPLE_INTERLOCK_GPIO_Port
 main.h, 206
SIMPLE_INTERLOCK_Pin
 main.h, 206
SOH

xmodem.h, 254
SPI, 14
 BLOCK_ERASE_32K, 14
 BLOCK_ERASE_4K, 14
 BLOCK_ERASE_64K, 15
 BYTE_PROGRAM, 15
 ERASE_CHIP1, 15
 ERASE_CHIP2, 15
 HIGH, 15
 LOW, 15
 READ_ARRAY1, 16
 READ_ARRAY2, 16
 READ_DEVICE_ID, 16
 READ_DUAL, 16
 READ_MANUFACTURE_ID, 16
 READ_QUAD, 16
 READ_STATUS_REGISTER1, 17
 READ_STATUS_REGISTER2, 17
 WRITE_DISABLE, 17
 WRITE_ENABLE, 17
 WRITE_ENABLE_STATUS_REGISTER_VOLATILE, status_register
 17
 WRITE_STATUS_REGISTER, 17
spi.c
 HAL_SPI_MspDeInit, 299
 HAL_SPI_MspInit, 300
 hspi1, 301
 MX_SPI1_Init, 300
spi.h
 hspi1, 225
 MX_SPI1_Init, 224
SRP0
 byte8_t_mem_status, 62
SRP1
 byte8_t_mem_status, 62
stall_pressure
 AdcController, 56
START_BEEP
 Auxilary, 21
start_motor
 PeripheralDevices, 147
State, 162
 ~State, 163
 battery, 164
 bypass, 164
 current, 164
 init_first_step, 163
 next, 164
 pressure, 165
 pulses_to_screen, 165
 State, 163
state
 main.cpp, 289
 MainController, 96
state.h
 Active, 227
 Idle, 227
 Init, 227
 QA, 227
 statesMachine, 227
 UartDebug, 227
 UpdateData, 227
 statesMachine
 state.h, 227
 status
 MainController, 96
 status1
 byte8_t_mem_status, 63
 status16_t
 Memory, 121
 status2
 byte8_t_mem_status, 63
 STATUS_reg, 165
 ~STATUS_reg, 167
 current, 168
 get_value, 167
 print_current, 167
 STATUS_reg, 167
 StatusReg
 PeripheralDevices, 150
stm32f0xx_hal_conf.h
 assert_param, 231
 DATA_CACHE_ENABLE, 231
 HAL_ADC_MODULE_ENABLED, 232
 HAL_CORTEX_MODULE_ENABLED, 232
 HAL_CRC_MODULE_ENABLED, 232
 HAL_DMA_MODULE_ENABLED, 232
 HAL_FLASH_MODULE_ENABLED, 232
 HAL_GPIO_MODULE_ENABLED, 232
 HAL_I2C_MODULE_ENABLED, 233
 HAL_MODULE_ENABLED, 233
 HAL_PWR_MODULE_ENABLED, 233
 HAL_RCC_MODULE_ENABLED, 233
 HAL_RTC_MODULE_ENABLED, 233
 HAL_SPI_MODULE_ENABLED, 233
 HAL_TIM_MODULE_ENABLED, 234
 HAL_UART_MODULE_ENABLED, 234
 HSE_STARTUP_TIMEOUT, 234
 HSE_VALUE, 234
 HSI14_VALUE, 234
 HSI48_VALUE, 235
 HSI_STARTUP_TIMEOUT, 235
 HSI_VALUE, 235
 INSTRUCTION_CACHE_ENABLE, 235
 LSE_STARTUP_TIMEOUT, 236
 LSE_VALUE, 236
 LSI_VALUE, 236
 PREFETCH_ENABLE, 236
 TICK_INT_PRIORITY, 236
 USERTOS, 237
 USESPI_CRC, 237
 VDD_VALUE, 237
stm32f0xx_hal_msp.c
 HAL_MspInit, 304

stm32f0xx_hal_timebase_tim.c
 HAL_InitTick, 306
 HAL_ResumeTick, 307
 HAL_SuspendTick, 307
 htim1, 308
 stm32f0xx_it.c
 DMA1_Channel1_IRQHandler, 310
 HardFault_Handler, 310
 hdma_adc, 314
 htim1, 314
 htim16, 314
 huart1, 314
 NMI_Handler, 310
 PendSV_Handler, 310
 read_reset_button, 311
 reset_pressed, 315
 SVC_Handler, 311
 SysTick_Handler, 311
 test_for_reset_press, 312
 TIM16_IRQHandler, 312
 TIM1_BRK_UP_TRG_COM_IRQHandler, 313
 USART1_IRQHandler, 313
 stm32f0xx_it.h
 DMA1_Channel1_IRQHandler, 239
 HardFault_Handler, 239
 NMI_Handler, 239
 PendSV_Handler, 240
 SVC_Handler, 240
 SysTick_Handler, 240
 TIM16_IRQHandler, 241
 TIM1_BRK_UP_TRG_COM_IRQHandler, 241
 USART1_IRQHandler, 242
 Stm32f0xx_system, 37
 STM32F0xx_System_Private_Defines, 40
 HSE_VALUE, 40
 HSI48_VALUE, 40
 HSI_VALUE, 40
 STM32F0xx_System_Private_FunctionPrototypes, 43
 STM32F0xx_System_Private_Functions, 44
 SystemCoreClockUpdate, 44
 SystemInit, 45
 STM32F0xx_System_Private_Includes, 38
 STM32F0xx_System_Private_Macros, 41
 STM32F0xx_System_Private_TypesDefinitions, 39
 STM32F0xx_System_Private_Variables, 42
 AHBPrescTable, 42
 APBPrescTable, 42
 SystemCoreClock, 42
 STM32L0xx_IAP, 31
 aFileName, 34
 dump_mem, 31
 FLASH_If_Erase, 31
 FLASH_If_Write, 32
 FlashProtection, 35
 JumpAddress, 35
 JumpToApplication, 35
 Main_Menu, 33
 memory_extern, 35
 STM32L0xx_IAP_Main, 27
 Int2Str, 27
 Serial_PutByte, 28
 Serial_PutString, 28
 Str2Int, 29
 stop_motor
 PeripheralDevices, 148
 Str2Int
 STM32L0xx_IAP_Main, 29
 STX
 xmodem.h, 254
 SVC_Handler
 stm32f0xx_it.c, 311
 stm32f0xx_it.h, 240
 SWEEP_SEQ_SELECT
 adc_command, 50
 SWITCHES_STATUS
 byte8_t_reg_status, 71
 SYS_SWO_RESERVED_GPIO_Port
 main.h, 207
 SYS_SWO_RESERVED_Pin
 main.h, 207
 SystemClock_Config
 main.cpp, 286
 SystemCoreClock
 STM32F0xx_System_Private_Variables, 42
 SystemCoreClockUpdate
 STM32F0xx_System_Private_Functions, 44
 SystemInit
 STM32F0xx_System_Private_Functions, 45
 SysTick_Handler
 stm32f0xx_it.c, 311
 stm32f0xx_it.h, 240
 TB
 byte8_t_mem_status, 63
 TECHNICIAN_ENABLED
 Globals, 13
 Temp
 adc_controller.h, 174
 temperature_mcu
 AdcController, 56
 Temperature_register_add
 Auxiliary, 21
 test_for_reset_press
 stm32f0xx_it.c, 312
 TICK_INT_PRIORITY
 stm32f0xx_hal_conf.h, 236
 TICKS_FOR_UPDATE
 Auxiliary, 21
 TICKS_SCREEN_UPDATE_LONG
 Auxiliary, 21
 TICKS_SCREEN_UPDATE_SUPER_LONG
 Auxiliary, 22
 tim.c
 HAL_TIM_Base_MspDelinit, 318
 HAL_TIM_Base_MspInit, 318
 htim16, 319
 MX_TIM16_Init, 318

tim.h
 htim16, 244
 MX_TIM16_Init, 243

TIM16_IRQHandler
 stm32f0xx_it.c, 312
 stm32f0xx_it.h, 241

TIM1_BRK_UP_TRG_COM_IRQHandler
 stm32f0xx_it.c, 313
 stm32f0xx_it.h, 241

TIMEOUT_SCREEN
 Auxiliary, 22

timer_last_update
 main.cpp, 290

TOGGLE400_RESET_BTN_GPIO_Port
 main.h, 207

TOGGLE400_RESET_BTN_Pin
 main.h, 207

toggle_motor
 PeripheralDevices, 149

tx_send_general_error
 Screen_ctrl, 155

TX_TIMEOUT
 common.h, 178

UART1_PC_RX_GPIO_Port
 main.h, 207

UART1_PC_RX_Pin
 main.h, 207

UART1_PC_TX_GPIO_Port
 main.h, 208

UART1_PC_TX_Pin
 main.h, 208

UART2_SCREEN_RX_GPIO_Port
 main.h, 208

UART2_SCREEN_RX_Pin
 main.h, 208

UART2_SCREEN_TX_GPIO_Port
 main.h, 208

UART2_SCREEN_TX_Pin
 main.h, 208

uart_state
 MainController, 90

UartDebug
 state.h, 227

Undefined
 adc_controller.h, 174

update_battery
 Screen_ctrl, 156

update_error_remaining
 Screen_ctrl, 157

update_piazo
 Screen_ctrl, 158

update_pressure
 Screen_ctrl, 159

update_pulse_counter
 Screen_ctrl, 160

update_remaining_in_percent
 Screen_ctrl, 160

update_state
 MainController, 91

UpdateData
 state.h, 227

usart.c
 HAL_UART_MspDeInit, 321
 HAL_UART_MspInit, 321
 huart1, 324
 huart2, 324
 MX_USART1_UART_Init, 322
 MX_USART2_UART_Init, 323

usart.h
 huart1, 247
 huart2, 248
 MX_USART1_UART_Init, 246
 MX_USART2_UART_Init, 246

USART1_IRQHandler
 stm32f0xx_it.c, 313
 stm32f0xx_it.h, 242

USB_DM_RESERVED_GPIO_Port
 main.h, 209

USB_DM_RESERVED_Pin
 main.h, 209

USB_DP_RESERVED_GPIO_Port
 main.h, 209

USB_DP_RESERVED_Pin
 main.h, 209

USE_RTOS
 stm32f0xx_hal_conf.h, 237

USE_SPI_CRC
 stm32f0xx_hal_conf.h, 237

USER_FLASH_SIZE
 flash_if.h, 187

V12
 adc_controller.h, 174

V5
 adc_controller.h, 174

V_Motor
 adc_controller.h, 174

value
 adc_command, 50
 bypass_state, 58
 byte8_t_mem_status, 63
 byte8_t_reg_cs, 66
 byte8_t_reg_readback, 68
 byte8_t_reg_status, 72
 channel, 73
 Monitor, 142

vbat
 AdcController, 56

VDD_sense
 adc_controller.h, 174

VDD_VALUE
 stm32f0xx_hal_conf.h, 237

VERSION
 Globals, 13

vref
 AdcController, 56

WATCHDOG_OUT_GPIO_Port
 main.h, 209
 WATCHDOG_OUT_Pin
 main.h, 209
 WEL
 byte8_t_mem_status, 63
 WIP
 byte8_t_mem_status, 63
 write
 Mem_ctrl, 107
 Memory, 118
 WRITE_CFR
 ADC, 26
 WRITE_DISABLE
 SPI, 17
 write_disable
 Memory, 118
 WRITE_ENABLE
 SPI, 17
 write_enable
 Memory, 119
 WRITE_ENABLE_STATUS_REGISTER_VOLATILE
 SPI, 17
 write_register
 Mem_ctrl, 109
 WRITE_STATUS_REGISTER
 SPI, 17
 write_status_register
 Memory, 119

 X_ACK
 xmodem.h, 255
 X_C
 xmodem.h, 255
 X_CAN
 xmodem.h, 255
 X_EOT
 xmodem.h, 255
 X_ERROR
 xmodem.h, 257
 X_ERROR_CRC
 xmodem.h, 257
 X_ERROR_FLASH
 xmodem.h, 257
 X_ERROR_NUMBER
 xmodem.h, 257
 X_ERROR_UART
 xmodem.h, 257
 X_MAX_ERRORS
 xmodem.h, 255
 X_NAK
 xmodem.h, 256
 X_OK
 xmodem.h, 257
 X_PACKET_1024_SIZE
 xmodem.h, 256
 X_PACKET_128_SIZE
 xmodem.h, 256
 X_PACKET_CRC_SIZE
 xmodem.h, 256

 xmodem.h, 256
 X_PACKET_DATA_INDEX
 xmodem.h, 256
 X_PACKET_NUMBER_COMPLEMENT_INDEX
 xmodem.h, 256
 X_PACKET_NUMBER_INDEX
 xmodem.h, 257
 X_SOH
 xmodem.h, 257
 X_STX
 xmodem.h, 257
 xmodem.cpp
 aPacketData, 328
 xmodem_calc_crc, 326
 xmodem_receive, 326

 xmodem.h
 ABORT1, 251
 ABORT2, 251
 ACK, 251
 CA, 251
 CRC16, 251
 DOWNLOAD_TIMEOUT, 251
 EOT, 252
 FILE_NAME_LENGTH, 252
 FILE_SIZE_LENGTH, 252
 MAX_ERRORS, 252
 NAK, 252
 NAK_TIMEOUT, 252
 NEGATIVE_BYTE, 253
 PACKET_1K_SIZE, 253
 PACKET_CNUMBER_INDEX, 253
 PACKET_DATA_INDEX, 253
 PACKET_HEADER_SIZE, 253
 PACKET_NUMBER_INDEX, 253
 PACKET_OVERHEAD_SIZE, 254
 PACKET_SIZE, 254
 PACKET_START_INDEX, 254
 PACKET_TRAILER_SIZE, 254
 SOH, 254
 STX, 254
 X_ACK, 255
 X_C, 255
 X_CAN, 255
 X_EOT, 255
 X_ERROR, 257
 X_ERROR_CRC, 257
 X_ERROR_FLASH, 257
 X_ERROR_NUMBER, 257
 X_ERROR_UART, 257
 X_MAX_ERRORS, 255
 X_NAK, 256
 X_OK, 257
 X_PACKET_1024_SIZE, 256
 X_PACKET_128_SIZE, 256
 X_PACKET_CRC_SIZE, 256
 X_PACKET_DATA_INDEX, 256
 X_PACKET_NUMBER_COMPLEMENT_INDEX,
 256

X_PACKET_NUMBER_INDEX, [257](#)
X_SOH, [257](#)
X_STX, [257](#)
xmodem_calc_crc, [258](#)
xmodem_receive, [258](#)
xmodem_status, [257](#)
xmodem_calc_crc
 xmodem.cpp, [326](#)
 xmodem.h, [258](#)
xmodem_receive
 xmodem.cpp, [326](#)
 xmodem.h, [258](#)
xmodem_status
 xmodem.h, [257](#)

Y0
 cs_register.h, [182](#)