RS485 SERVER

V0.3

Generated by Doxygen 1.8.8

Fri Apr 22 2016 15:56:03

Contents

1 RS485 SERVER API Documentation **1**

2 Module Index **3**

2.1 Modules 3

3 Data Structure Index **5**

3.1 Data Structures 5

4 File Index **7**

4.1 File List 7

5 Module Documentation **9**

5.1 Management 9

5.1.1 Detailed Description 10

5.2 Protocol 11

5.2.1 Detailed Description 11

5.3 BACnet 12

5.3.1 Detailed Description 12

5.4 Modbus 13

5.4.1 Detailed Description 13

5.5 General 14

5.5.1 Detailed Description 14

5.6 Device 15

5.6.1 Detailed Description 15

5.7 AirConditon 16

5.7.1 Detailed Description 16

5.8 Curtain 17

5.8.1 Detailed Description 17

5.9 FreshAir 18

5.9.1 Detailed Description 18

5.10 Adapter management 19

5.10.1 Detailed Description 20

5.10.2 Function Documentation 20

5.10.2.1 adapter_thread_init 20
5.10.2.2 process_read_value_service 20
5.10.2.3 process_write_value_service 21
5.11 DaiKin DTA116A621 23
5.11.1 Detailed Description 23
5.11.2 Function Documentation 23
5.11.2.1 daikin_dta116a621_get_device_info_handle 23
5.11.2.2 daikin_dta116a621_get_device_info_send 24
5.11.2.3 daikin_dta116a621_set_fan 25
5.11.2.4 daikin_dta116a621_set_mode 25
5.11.2.5 daikin_dta116a621_set_swing 26
5.11.2.6 daikin_dta116a621_set_switch 26
5.11.2.7 daikin_dta116a621_set_temperature 27
5.12 Panasonnic GuangZhou 28
5.12.1 Detailed Description 28
5.12.2 Function Documentation 28
5.12.2.1 panasonnic_recv_package_handle 28
5.12.2.2 panasonnic_send_package_handle 28
5.13 YORK GuangZhou KeLong 30
5.13.1 Detailed Description 30
5.13.2 Function Documentation 30
5.13.2.1 get_air_york_instance 30
5.13.2.2 get_air_york_read_args 30
5.13.2.3 get_air_york_write_args 31
5.14 Aoke GuangZhou 32
5.14.1 Detailed Description 32
5.14.2 Function Documentation 32
5.14.2.1 aoke_recv_package_handle 32
5.14.2.2 aoke_send_package_handle 32
5.15 Dooya HangZhou 34
5.15.1 Detailed Description 34
5.15.2 Function Documentation 34
5.15.2.1 doya_recv_package_handle 34
5.15.2.2 doya_send_package_handle 35
5.16 Loreley ShenZhen 37
5.16.1 Detailed Description 37
5.16.2 Function Documentation 37
5.16.2.1 loreley_recv_package_handle 37
5.16.2.2 loreley_send_package_handle 38
5.17 Device management 40

5.17.1 Detailed Description 41
5.17.2 Typedef Documentation 41
5.17.2.1 device_management_t 41
5.17.3 Function Documentation 41
5.17.3.1 check_device_id 41
5.17.3.2 create_device 42
5.17.3.3 delete_device 43
5.17.3.4 device_managemnt_init 44
5.17.3.5 find_available_device_id 45
5.17.3.6 get_device_addr 45
5.17.3.7 get_device_address_len 45
5.17.3.8 get_device_factory_name 46
5.17.3.9 get_device_management 46
5.17.3.10 get_device_name 47
5.17.3.11 get_device_object_id 48
5.17.3.12 get_device_private 48
5.17.3.13 get_device_private_numbers 49
5.17.3.14 get_device_protocol 49
5.17.3.15 get_device_retransmission 50
5.17.3.16 get_device_timeout_ms 50
5.17.3.17 get_device_timer 51
5.17.3.18 get_device_type 51
5.17.3.19 get_read_device_information 52
5.17.3.20 set_read_device_information 53
5.18 String management 55
5.18.1 Detailed Description 55
5.18.2 Function Documentation 55
5.18.2.1 get_enum_txt_bool 55
5.18.2.2 get_enum_txt_device_factory 56
5.18.2.3 get_enum_txt_device_method 56
5.18.2.4 get_enum_txt_rs485_device_type 56
5.18.2.5 get_enum_txt_rs485_protocol_type 57
5.18.2.6 get_enum_txt_service 57
5.19 Item management 58
5.19.1 Detailed Description 58
5.19.2 Macro Definition Documentation 58
5.19.2.1 PANNO_S_ITEM_ARMANI 58
5.19.2.2 PANNO_S_ITEM_CONFIG 58
5.19.2.3 PANNO_S_ITEM_DEFAULT 59
5.19.2.4 PANNO_S_ITEM_SHAOCHENGGUOJI 59

5.19.2.5 PANNO_S_ITEM_WENRUDE 59
5.19.3 Function Documentation 59
5.19.3.1 panno_s_item_config 59
5.20 Object management 60
5.20.1 Detailed Description 61
5.20.2 Typedef Documentation 61
5.20.2.1 object_management_t 61
5.20.3 Function Documentation 61
5.20.3.1 check_object_id 61
5.20.3.2 check_object_numbers_have_idle 62
5.20.3.3 create_object 63
5.20.3.4 delete_object 64
5.20.3.5 find_available_object_id 65
5.20.3.6 get_object_mount_device 65
5.20.3.7 get_object_queue_sem 66
5.20.3.8 get_object_type 67
5.20.3.9 get_object_work_queue 68
5.20.3.10 object_is_used 68
5.20.3.11 object_mount_device_id 69
5.20.3.12 object_unmount_device_id 70
5.20.3.13 work_thread_clean 71
5.20.3.14 work_thread_create 71
5.21 BACnet interface 73
5.21.1 Detailed Description 73
5.21.2 Typedef Documentation 73
5.21.2.1 bacnet_port_handle_t 73
5.21.3 Function Documentation 73
5.21.3.1 bacnet_work_thread_function 73
5.22 Handle_property 75
5.22.1 Detailed Description 75
5.22.2 Macro Definition Documentation 75
5.22.2.1 BACNET_READ_ARGS_OBJECT_MAX 75
5.22.3 Function Documentation 75
5.22.3.1 bacnet_service_init 75
5.22.3.2 get_air_condition_bacnet_read_args 76
5.22.3.3 get_air_condition_bacnet_write_args 77
5.23 General interface 79
5.23.1 Detailed Description 79
5.23.2 Typedef Documentation 79
5.23.2.1 mstp_port_handle_t 79

5.23.3 Function Documentation 79
5.23.3.1 general_service_init 79
5.23.3.2 general_work_thread_function 80
5.24 General RS485 82
5.24.1 Detailed Description 82
5.24.2 Function Documentation 82
5.24.2.1 rs485_cleanup 82
5.24.2.2 rs485_get_interface 83
5.24.2.3 rs485_initialize 83
5.24.2.4 rs485_recv_handle_frame 83
5.24.2.5 rs485_send_handle_frame 84
5.24.2.6 rs485_set_baud_rate 85
5.24.2.7 rs485_set_interface 85
5.25 Modbus RS485 86
5.25.1 Detailed Description 86
5.25.2 Function Documentation 86
5.25.2.1 modbus_service_deinit 86
5.25.2.2 modbus_service_init 87
5.25.2.3 modbus_work_thread_function 88
5.26 Service management 89
5.26.1 Detailed Description 89
5.26.2 Function Documentation 89
5.26.2.1 rs485_recv_msg_from_client 89
5.26.2.2 rs485_send_msg_to_client 90
5.26.2.3 rs485_service_create_clean 90
5.26.2.4 rs485_service_running 90
5.26.2.5 rs485_service_running_clean 91
5.26.2.6 rs485_service_start 92
5.26.2.7 rs485_thread_pool_clean 92
5.26.2.8 rs485_thread_pool_create 93
5.26.2.9 send_msg_to_adapter 93
5.27 Device register management 94
5.27.1 Detailed Description 94
5.27.2 Typedef Documentation 94
5.27.2.1 method_recv 94
5.27.2.2 method_send 95
5.27.3 Function Documentation 95
5.27.3.1 check_device_support 95
5.27.3.2 get_device_recv_package_function 96
5.27.3.3 get_device_send_package_function 96

5.27.3.4 get_support_device_profile . . . 96
5.27.3.5 get_support_device_profile_numbers . . . 97
5.28 Timer management . . . 98
5.28.1 Detailed Description . . . 98
5.28.2 Typedef Documentation . . . 98
5.28.2.1 timer_proc_func . . . 98
5.28.3 Function Documentation . . . 98
5.28.3.1 create_device_timer_task . . . 98
5.28.3.2 delete_device_timer_task . . . 99
5.28.3.3 device_timer_task_handle_demo . . . 99
5.28.3.4 timer_task_thread_function . . . 100

6 Data Structure Documentation 101

6.1 adapter_t Struct Reference . . . 101
6.1.1 Detailed Description . . . 101
6.1.2 Field Documentation . . . 102
6.1.2.1 message_content . . . 102
6.1.2.2 message_length . . . 102
6.1.2.3 message_retvl . . . 102
6.1.2.4 message_type . . . 102
6.1.2.5 socket_fd . . . 102
6.2 air_condition_profile_t Struct Reference . . . 102
6.2.1 Detailed Description . . . 103
6.2.2 Field Documentation . . . 103
6.2.2.1 current_fan . . . 103
6.2.2.2 current_mode . . . 103
6.2.2.3 current_set_temperature . . . 103
6.2.2.4 current_swing . . . 103
6.2.2.5 outdoor_temperature . . . 103
6.2.2.6 pipe_temperature . . . 103
6.2.2.7 room_temperature . . . 103
6.3 bacnet Struct Reference . . . 103
6.3.1 Detailed Description . . . 104
6.3.2 Field Documentation . . . 104
6.3.2.1 arg . . . 104
6.3.2.2 command . . . 104
6.3.2.3 device_mac . . . 104
6.3.2.4 device_type . . . 104
6.3.2.5 factory_name . . . 104
6.3.2.6 mac_length . . . 104

6.3.2.7 value . . . 104
6.3.2.8 value_reserve . . . 104
6.4 bacnet_read_args_t Struct Reference . . . 105
6.4.1 Detailed Description . . . 105
6.4.2 Field Documentation . . . 105
6.4.2.1 device_id . . . 105
6.4.2.2 object_instance . . . 105
6.4.2.3 object_numbers . . . 105
6.4.2.4 object_property . . . 105
6.4.2.5 object_type . . . 105
6.5 bacnet_write_args_t Struct Reference . . . 106
6.5.1 Detailed Description . . . 106
6.5.2 Field Documentation . . . 106
6.5.2.1 device_id . . . 106
6.5.2.2 object_instance . . . 106
6.5.2.3 object_property . . . 106
6.5.2.4 object_property_index . . . 106
6.5.2.5 object_property_priority . . . 106
6.5.2.6 object_property_value . . . 107
6.5.2.7 object_property_value_type . . . 107
6.5.2.8 object_type . . . 107
6.6 client_t Struct Reference . . . 107
6.6.1 Detailed Description . . . 107
6.6.2 Field Documentation . . . 107
6.6.2.1 fd . . . 107
6.6.2.2 uid . . . 107
6.7 commonBacObj_s Struct Reference . . . 107
6.7.1 Detailed Description . . . 108
6.7.2 Field Documentation . . . 108
6.7.2.1 mObject_Type . . . 108
6.7.2.2 Object_Instance_Number . . . 108
6.7.2.3 Object_Name . . . 108
6.8 create_object_return_t Struct Reference . . . 108
6.8.1 Detailed Description . . . 108
6.8.2 Field Documentation . . . 109
6.8.2.1 object_id . . . 109
6.9 create_object_t Struct Reference . . . 109
6.9.1 Detailed Description . . . 109
6.9.2 Field Documentation . . . 109
6.9.2.1 address . . . 109

6.9.2.2 address_len . . . 110
6.9.2.3 mount_device_max . . . 110
6.9.2.4 object_name . . . 110
6.9.2.5 object_type . . . 110
6.9.2.6 port . . . 110
6.10 curtain_profile_t Struct Reference . . . 110
6.10.1 Detailed Description . . . 110
6.10.2 Field Documentation . . . 111
6.10.2.1 current_percent . . . 111
6.11 delete_object_return_t Struct Reference . . . 111
6.11.1 Detailed Description . . . 111
6.11.2 Field Documentation . . . 111
6.11.2.1 delete_status . . . 111
6.12 delete_object_t Struct Reference . . . 111
6.12.1 Detailed Description . . . 111
6.12.2 Field Documentation . . . 112
6.12.2.1 object_id . . . 112
6.13 device_management Struct Reference . . . 112
6.13.1 Detailed Description . . . 113
6.13.2 Field Documentation . . . 113
6.13.2.1 device . . . 113
6.13.2.2 device_addr . . . 113
6.13.2.3 device_addr_len . . . 113
6.13.2.4 device_id . . . 113
6.13.2.5 device_info . . . 113
6.13.2.6 device_name . . . 113
6.13.2.7 device_private . . . 113
6.13.2.8 device_private_numbers . . . 114
6.13.2.9 device_status_period . . . 114
6.13.2.10 device_type . . . 114
6.13.2.11 factory_name . . . 114
6.13.2.12 object_id . . . 114
6.13.2.13 object_type . . . 114
6.13.2.14 retransmission . . . 114
6.13.2.15 support_reply . . . 114
6.13.2.16 time_out . . . 114
6.13.2.17 timer . . . 115
6.14 device_profile Struct Reference . . . 115
6.14.1 Detailed Description . . . 115
6.14.2 Field Documentation . . . 115

6.14.2.1 addr_real_len 115
6.14.2.2 method 115
6.14.2.3 recv 115
6.14.2.4 send 115
6.15 devObj_s Struct Reference 116
6.15.1 Detailed Description 116
6.15.2 Field Documentation 116
6.15.2.1 bacDevAddr 116
6.15.2.2 bacObj 116
6.15.2.3 Database_Revision 116
6.15.2.4 Description 117
6.16 fresh_air_profile_t Struct Reference 117
6.16.1 Detailed Description 117
6.16.2 Field Documentation 117
6.16.2.1 fresh_level 117
6.16.2.2 pm2_5 117
6.16.2.3 room_humidity 117
6.16.2.4 room_temperature 117
6.17 message_service_t Union Reference 118
6.17.1 Detailed Description 118
6.17.2 Field Documentation 118
6.17.2.1 delete_object 118
6.17.2.2 delete_object_return 119
6.17.2.3 mount_device 119
6.17.2.4 mount_device_return 119
6.17.2.5 new_object 119
6.17.2.6 new_object_return 119
6.17.2.7 read 119
6.17.2.8 read_return 119
6.17.2.9 unmount_device 119
6.17.2.10 unmount_device_return 119
6.17.2.11 write 119
6.17.2.12 write_return 119
6.18 modbus_port_handle_t Struct Reference 120
6.18.1 Detailed Description 120
6.18.2 Field Documentation 120
6.18.2.1 broadcast 120
6.18.2.2 buffer 120
6.18.2.3 buffer_len 120
6.18.2.4 code 120

6.18.2.5 device_addr 121
6.18.2.6 device_id 121
6.18.2.7 method 121
6.18.2.8 recv_handle 121
6.18.2.9 register_addr 121
6.18.2.10 retransmission 121
6.18.2.11 send_handle 121
6.18.2.12 value 121
6.19 mount_devcie_to_object_t Struct Reference 121
6.19.1 Detailed Description 122
6.19.2 Field Documentation 122
6.19.2.1 device_addr 122
6.19.2.2 device_addr_len 122
6.19.2.3 device_name 122
6.19.2.4 device_status_period 122
6.19.2.5 device_type 122
6.19.2.6 factory_name 122
6.19.2.7 object_id 123
6.19.2.8 object_type 123
6.19.2.9 retransmission 123
6.19.2.10 support_reply 123
6.19.2.11 time_out 123
6.20 mount_device_to_object_return_t Struct Reference 123
6.20.1 Detailed Description 123
6.20.2 Field Documentation 123
6.20.2.1 device_id 123
6.21 mstp_port_handle Struct Reference 124
6.21.1 Detailed Description 124
6.21.2 Field Documentation 124
6.21.2.1 address 124
6.21.2.2 address_len 124
6.21.2.3 arg 124
6.21.2.4 broadcast 125
6.21.2.5 device_id 125
6.21.2.6 except_reply 125
6.21.2.7 method 125
6.21.2.8 package_buffer 125
6.21.2.9 package_buffer_len 125
6.21.2.10 recv_handle 125
6.21.2.11 retransmission 125

6.21.2.12 send_handle 125
6.21.2.13 timeout_ms 126
6.21.2.14 value 126
6.22 object_functions Struct Reference 126
6.22.1 Detailed Description 126
6.22.2 Field Documentation 126
6.22.2.1 Object_Count 126
6.22.2.2 Object_COV 127
6.22.2.3 Object_COV_Clear 127
6.22.2.4 Object_Index_To_Instance 127
6.22.2.5 Object_Init 127
6.22.2.6 Object_Intrinsic_Reporting 127
6.22.2.7 Object_Iterator 127
6.22.2.8 Object_Name 127
6.22.2.9 Object_Read_Property 127
6.22.2.10 Object_RPM_List 127
6.22.2.11 Object_RR_Info 127
6.22.2.12 Object_Type 127
6.22.2.13 Object_Valid_Instance 127
6.22.2.14 Object_Value_List 128
6.22.2.15 Object_Write_Property 128
6.23 object_management Struct Reference 128
6.23.1 Detailed Description 129
6.23.2 Field Documentation 129
6.23.2.1 address 129
6.23.2.2 address_len 129
6.23.2.3 mount_device 129
6.23.2.4 mount_device_max 129
6.23.2.5 object_id 129
6.23.2.6 object_name 129
6.23.2.7 object_private 129
6.23.2.8 object_thread 129
6.23.2.9 object_type 130
6.23.2.10 port 130
6.23.2.11 queue_depth 130
6.23.2.12 queue_sem 130
6.23.2.13 work_queue 130
6.23.2.14 work_queue_buffer 130
6.24 package Struct Reference 130
6.24.1 Detailed Description 130

6.24.2 Field Documentation . . . 131
6.24.2.1 addr_high . . . 131
6.24.2.2 addr_low . . . 131
6.24.2.3 cmd . . . 131
6.24.2.4 command . . . 131
6.24.2.5 data . . . 131
6.24.2.6 data_addr . . . 131
6.25 read_device_return_t Struct Reference . . . 131
6.25.1 Detailed Description . . . 132
6.25.2 Field Documentation . . . 132
6.25.2.1 error . . . 132
6.25.2.2 profile . . . 132
6.25.2.3 read_status . . . 132
6.25.2.4 runing . . . 133
6.26 read_device_t Struct Reference . . . 133
6.26.1 Detailed Description . . . 133
6.26.2 Field Documentation . . . 133
6.26.2.1 device_id . . . 133
6.27 rs485_curtain_ao_ke_send_package_t Struct Reference . . . 133
6.27.1 Detailed Description . . . 133
6.27.2 Field Documentation . . . 134
6.27.2.1 d1 . . . 134
6.27.2.2 d2 . . . 134
6.27.2.3 d3 . . . 134
6.27.2.4 d4 . . . 134
6.27.2.5 d5 . . . 134
6.28 rs485_device_profile Union Reference . . . 134
6.28.1 Detailed Description . . . 135
6.28.2 Field Documentation . . . 135
6.28.2.1 air_condition . . . 135
6.28.2.2 curtain . . . 135
6.28.2.3 fresh_air . . . 135
6.29 rs485_port_t Struct Reference . . . 135
6.29.1 Detailed Description . . . 135
6.29.2 Field Documentation . . . 135
6.29.2.1 baud_rate . . . 135
6.29.2.2 interface_name . . . 136
6.30 thread_pool_t Struct Reference . . . 136
6.30.1 Detailed Description . . . 136
6.30.2 Field Documentation . . . 136

6.30.2.1 arg 136
6.30.2.2 attr 136
6.30.2.3 function 136
6.30.2.4 thread 136
6.30.2.5 thread_status 137
6.31 timer_task_t Struct Reference 137
6.31.1 Detailed Description 137
6.31.2 Field Documentation 137
6.31.2.1 command 137
6.31.2.2 device_id 137
6.31.2.3 function 137
6.31.2.4 tick 137
6.31.2.5 timeout 138
6.32 unmount_device_from_object_return_t Struct Reference 138
6.32.1 Detailed Description 138
6.32.2 Field Documentation 138
6.32.2.1 unmount_status 138
6.33 unmount_device_from_object_t Struct Reference 138
6.33.1 Detailed Description 138
6.33.2 Field Documentation 139
6.33.2.1 device_id 139
6.33.2.2 object_id 139
6.34 write_device_return_t Struct Reference 139
6.34.1 Detailed Description 139
6.34.2 Field Documentation 139
6.34.2.1 write_status 139
6.35 write_device_t Struct Reference 139
6.35.1 Detailed Description 140
6.35.2 Field Documentation 140
6.35.2.1 broadcast 140
6.35.2.2 device_id 140
6.35.2.3 device_method 140
6.35.2.4 method_value 140
6.35.2.5 value_reserve 140

7 File Documentation 141
7.1 include/adapter.h File Reference 141
7.1.1 Detailed Description 142
7.2 include/device.h File Reference 143
7.2.1 Detailed Description 144

7.3 include/device/airCondition/daikin/DTA116A621.h File Reference 145
7.4 include/device/airCondition/panasonnic/panasonnic.h File Reference 146
7.5 include/device/airCondition/york/york.h File Reference 146
7.6 include/device/curtain/aoke/aoke.h File Reference 147
7.7 include/device/curtain/doya/doya.h File Reference 148
7.8 include/device/freshAir/loreley/loreley.h File Reference 149
7.9 include/enum.h File Reference 150
7.9.1 Detailed Description 152
7.9.2 Macro Definition Documentation 152
7.9.2.1 UNUSED 152
7.9.3 Enumeration Type Documentation 152
7.9.3.1 adapter_thread_status_enum 152
7.9.3.2 modbus_function_code_enum 153
7.9.3.3 object_thread_status_enum 153
7.9.3.4 rs485_device_method_enum 153
7.9.3.5 rs485_device_type_enum 155
7.9.3.6 rs485_factory_name_enum 155
7.9.3.7 rs485_method_air_condition_panasonnic_enum 155
7.9.3.8 rs485_method_air_condition_york_enum 156
7.9.3.9 rs485_method_curtain_aoke_enum 158
7.9.3.10 rs485_method_curtain_doya_enum 158
7.9.3.11 rs485_method_fresh_air_loreley_enum 158
7.9.3.12 rs485_protocol_type_enum 158
7.9.3.13 rs485_service_type_enum 159
7.9.3.14 timer_task_thread_status_enum 159
7.10 include/enumtxt.h File Reference 159
7.11 include/item_config.h File Reference 160
7.12 include/object.h File Reference 162
7.13 include/protocol/bacnet/bacnet.h File Reference 163
7.14 include/protocol/bacnet/device_client.h File Reference 164
7.14.1 Detailed Description 167
7.14.2 Macro Definition Documentation 167
7.14.2.1 MAX_DEV_DESC_LEN 167
7.14.2.2 MAX_DEV_LOC_LEN 167
7.14.2.3 MAX_DEV_MOD_LEN 167
7.14.2.4 MAX_DEV_NAME_LEN 167
7.14.2.5 MAX_DEV_VER_LEN 167
7.14.3 Typedef Documentation 167
7.14.3.1 COMMON_BAC_OBJECT 167
7.14.3.2 DEVICE_OBJECT_DATA 167

7.14.3.3 object_count_function 167
7.14.3.4 object_cov_clear_function 168
7.14.3.5 object_cov_function 169
7.14.3.6 object_functions_t 169
7.14.3.7 object_index_to_instance_function 169
7.14.3.8 object_init_function 169
7.14.3.9 object_intrinsic_reporting_function 169
7.14.3.10 object_iterate_function 170
7.14.3.11 object_name_function 170
7.14.3.12 object_valid_instance_function 170
7.14.3.13 object_value_list_function 170
7.14.4 Function Documentation 171
7.14.4.1 Add_Routed_Device 171
7.14.4.2 Device_Application_Software_Version 171
7.14.4.3 Device_Count 171
7.14.4.4 Device_COV 171
7.14.4.5 Device_COV_Clear 171
7.14.4.6 Device_Database_Revision 171
7.14.4.7 Device_Description 171
7.14.4.8 Device_Encode_Value_List 171
7.14.4.9 Device_Firmware_Revision 171
7.14.4.10 Device_getCurrentDateTime 171
7.14.4.11 Device_Inc_Database_Revision 171
7.14.4.12 Device_Index_To_Instance 172
7.14.4.13 Device_Init 172
7.14.4.14 Device_Location 172
7.14.4.15 Device_Model_Name 172
7.14.4.16 Device_Object_Instance_Number 172
7.14.4.17 Device_Object_List_Count 173
7.14.4.18 Device_Object_List_Identifier 173
7.14.4.19 Device_Object_Name 174
7.14.4.20 Device_Object_Name_Copy 174
7.14.4.21 Device_Objects_Property_List 175
7.14.4.22 Device_Objects_RR_Info 175
7.14.4.23 Device_Property_Lists 175
7.14.4.24 Device_Protocol_Revision 175
7.14.4.25 Device_Protocol_Version 175
7.14.4.26 Device_Read_Property 175
7.14.4.27 Device_Read_Property_Local 176
7.14.4.28 Device_Reinitialize 176

7.14.4.29 Device_Reinitialized_State . . . 176
7.14.4.30 Device_Segmentation_Supported . . . 176
7.14.4.31 Device_Set_Application_Software_Version . . . 177
7.14.4.32 Device_Set_Database_Revision . . . 177
7.14.4.33 Device_Set_Description . . . 177
7.14.4.34 Device_Set_Location . . . 177
7.14.4.35 Device_Set_Model_Name . . . 177
7.14.4.36 Device_Set_Object_Instance_Number . . . 177
7.14.4.37 Device_Set_Object_Name . . . 178
7.14.4.38 Device_Set_System_Status . . . 178
7.14.4.39 Device_Set_Vendor_Identifier . . . 178
7.14.4.40 Device_System_Status . . . 178
7.14.4.41 Device_Valid_Object_Id . . . 178
7.14.4.42 Device_Valid_Object_Instance_Number . . . 179
7.14.4.43 Device_Valid_Object_Name . . . 179
7.14.4.44 Device_Value_List_Supported . . . 179
7.14.4.45 Device_Vendor_Identifier . . . 179
7.14.4.46 Device_Vendor_Name . . . 180
7.14.4.47 Device_Write_Property . . . 180
7.14.4.48 Device_Write_Property_Local . . . 180
7.14.4.49 DeviceGetRRInfo . . . 180
7.14.4.50 Get_Routed_Device_Address . . . 180
7.14.4.51 Get_Routed_Device_Object . . . 180
7.14.4.52 Routed_Device_Address_Lookup . . . 180
7.14.4.53 Routed_Device_GetNext . . . 180
7.14.4.54 Routed_Device_Inc_Database_Revision . . . 180
7.14.4.55 Routed_Device_Index_To_Instance . . . 180
7.14.4.56 Routed_Device_Is_Valid_Network . . . 180
7.14.4.57 Routed_Device_Name . . . 180
7.14.4.58 Routed_Device_Object_Instance_Number . . . 180
7.14.4.59 Routed_Device_Service_Approval . . . 180
7.14.4.60 Routed_Device_Set_Description . . . 180
7.14.4.61 Routed_Device_Set_Object_Instance_Number . . . 180
7.14.4.62 Routed_Device_Set_Object_Name . . . 180
7.14.4.63 Routed_Device_Valid_Object_Instance_Number . . . 180
7.14.4.64 routed_get_my_address . . . 180
7.14.4.65 Routing_Device_Init . . . 181
7.15 include/protocol/bacnet/handle_property.h File Reference . . . 181
7.16 include/protocol/bacnet/read_property.h File Reference . . . 182
7.16.1 Function Documentation . . . 183

7.16.1.1 bacnet_read_property . . . 183
7.17 include/protocol/bacnet/write_property.h File Reference . . . 183
7.17.1 Function Documentation . . . 184
7.17.1.1 bacnet_write_property . . . 184
7.18 include/protocol/general/general.h File Reference . . . 184
7.19 include/protocol/general/rs485.h File Reference . . . 185
7.20 include/rs485.h File Reference . . . 187
7.21 include/protocol/modbus/modbus.h File Reference . . . 187
7.22 include/read_config.h File Reference . . . 188
7.22.1 Variable Documentation . . . 188
7.22.1.1 glb_config_adapter_message_queue_depth . . . 188
7.22.1.2 glb_config_bacnet_work_queue_depth . . . 188
7.22.1.3 glb_config_general_work_package_mtu . . . 188
7.22.1.4 glb_config_general_work_queue_depth . . . 188
7.22.1.5 glb_config_modbus_work_queue_depth . . . 188
7.23 include/service.h File Reference . . . 189
7.23.1 Detailed Description . . . 190
7.24 include/support.h File Reference . . . 190
7.25 include/syslog/log.h File Reference . . . 191
7.25.1 Detailed Description . . . 192
7.25.2 Macro Definition Documentation . . . 192
7.25.2.1 DEBUG_LINUX_SYSLOG . . . 192
7.25.2.2 DEBUG_PRINTF . . . 192
7.25.2.3 syslog_debug . . . 192
7.25.2.4 syslog_error . . . 192
7.25.2.5 syslog_format . . . 192
7.25.2.6 syslog_info . . . 192
7.25.2.7 syslog_warning . . . 192
7.26 include/timer_task.h File Reference . . . 193
7.26.1 Detailed Description . . . 194
7.26.2 Function Documentation . . . 194
7.26.2.1 device_timer_task_handle_curtain_aoke_init . . . 194
7.26.2.2 device_timer_task_handle_curtain_doya_init . . . 194
7.26.2.3 device_timer_task_handle_curtain_init . . . 194
7.27 src/adapter.c File Reference . . . 194
7.27.1 Detailed Description . . . 195
7.27.2 Macro Definition Documentation . . . 195
7.27.2.1 ADAPTER_MESSAGE_QUEUE_MAX_DEPTH . . . 195
7.27.3 Function Documentation . . . 195
7.27.3.1 adapter_thread_function . . . 195

7.27.4 Variable Documentation 196
7.27.4.1 adapter_thread_status 196
7.27.4.2 bacnet 197
7.27.4.3 general 197
7.27.4.4 modbus 197
7.27.4.5 reply_client 197
7.28 src/device.c File Reference 197
7.28.1 Detailed Description 199
7.28.2 Macro Definition Documentation 199
7.28.2.1 RS485_CURTAIN_MAX_FACTORY 199
7.28.2.2 RS485_DEVICE_MAX_NUMBERS 199
7.28.2.3 RS485_DEVICE_NAME_MAX_LENGTH 199
7.28.3 Function Documentation 199
7.28.3.1 find_curtain_factory_is_save 199
7.28.3.2 printf_device 200
7.28.3.3 set_timer 200
7.28.4 Variable Documentation 201
7.28.4.1 curtain_factory 201
7.28.4.2 device_management_lock 201
7.28.4.3 glb_device_manage 201
7.29 src/device/airCondition/daikin/DTA116A621.c File Reference 202
7.29.1 Macro Definition Documentation 203
7.29.1.1 DAIKIN_HOLD_REGISTER_DEVICE_CONTROL_START 203
7.29.1.2 DAIKIN_HOLD_REGISTER_DEVICE_CONTROL_STOP 203
7.29.1.3 DAIKIN_HOLD_REGISTER_DEVICE_MODE_OFFSET 203
7.29.1.4 DAIKIN_HOLD_REGISTER_DEVICE_ON_OFF_SWING_FAN_OFFSET 203
7.29.1.5 DAIKIN_HOLD_REGISTER_DEVICE_TEMPERATURE_OFFSET 203
7.29.1.6 DAIKIN_HOLD_REGISTER_START 203
7.29.1.7 DAIKIN_INPUT_REGISTER_DEVICE_COMM_STATUS_START 203
7.29.1.8 DAIKIN_INPUT_REGISTER_DEVICE_COMM_STATUS_STOP 203
7.29.1.9 DAIKIN_INPUT_REGISTER_DEVICE_CONNECT_STATUS_START 204
7.29.1.10 DAIKIN_INPUT_REGISTER_DEVICE_CONNECT_STATUS_STOP 204
7.29.1.11 DAIKIN_INPUT_REGISTER_DEVICE_FUNCTION_STATUS_START 204
7.29.1.12 DAIKIN_INPUT_REGISTER_DEVICE_FUNCTION_STATUS_STOP 204
7.29.1.13 DAIKIN_INPUT_REGISTER_DEVICE_STATUS_START 204
7.29.1.14 DAIKIN_INPUT_REGISTER_DEVICE_STATUS_STOP 204
7.29.1.15 DAIKIN_INPUT_REGISTER_START 204
7.29.2 Function Documentation 204
7.29.2.1 get_daikin_hold_register_value 204
7.29.2.2 get_daikin_write_register_addr 205

7.29.3 Variable Documentation 206
7.29.3.1 glb_daikin_hold_register_value 206
7.30 src/device/airCondition/panasonnic/panasonnic.c File Reference 206
7.30.1 Macro Definition Documentation 208
7.30.1.1 ADR_BROADCAST 208
7.30.1.2 ADR_DEFAULT 208
7.30.1.3 ARG_ERROR 209
7.30.1.4 EOI 209
7.30.1.5 ERROR 209
7.30.1.6 GET_AIR_ADDR 209
7.30.1.7 GET_AIR_ADDR_CID_SEND 209
7.30.1.8 GET_AIR_INFO_CID_SEND 209
7.30.1.9 GET_AIR_INFO_ERROR_CODE_NORMAL 209
7.30.1.10 GET_AIR_INFO_OFF 210
7.30.1.11 GET_AIR_INFO_ON 210
7.30.1.12 GET_AIR_INFO_SET_TEMPERATURE 210
7.30.1.13 GET_AIR_INFO_SET_TEMPERATURE_ADD_1 210
7.30.1.14 GET_AIR_INFO_SET_TEMPERATURE_ADD_5 210
7.30.1.15 GET_AIR_INFO_SET_TEMPERATURE_DOWN 210
7.30.1.16 GET_AIR_INFO_SET_TEMPERATURE_DRY 210
7.30.1.17 GET_AIR_INFO_SET_TEMPERATURE_STD 210
7.30.1.18 GET_AIR_INFO_SET_TEMPERATURE_SUB_1 210
7.30.1.19 GET_AIR_INFO_SET_TEMPERATURE_SUB_5 210
7.30.1.20 GET_AIR_INFO_SET_TEMPERATURE_UP 210
7.30.1.21 GET_AIR_INFO_SET_WIND_DIRECTION_FOCUS 210
7.30.1.22 GET_AIR_INFO_SET_WIND_DIRECTION_HANDL1 211
7.30.1.23 GET_AIR_INFO_SET_WIND_DIRECTION_HANDL2 211
7.30.1.24 GET_AIR_INFO_SET_WIND_DIRECTION_HANDL3 211
7.30.1.25 GET_AIR_INFO_SET_WIND_DIRECTION_HANDL4 211
7.30.1.26 GET_AIR_INFO_SET_WIND_DIRECTION_HANDL5 211
7.30.1.27 GET_AIR_INFO_SET_WIND_DIRECTION_WIDE 211
7.30.1.28 GET_AIR_INFO_SET_WIND_RATE_AUTO 211
7.30.1.29 GET_AIR_INFO_SET_WIND_RATE_HIGH 211
7.30.1.30 GET_AIR_INFO_SET_WIND_RATE_LOW 211
7.30.1.31 GET_AIR_INFO_SET_WIND_RATE_MIDDLE 211
7.30.1.32 GET_AIR_INFO_SET_WIND_RATE_MOST 211
7.30.1.33 GET_AIR_INFO_SET_WIND_RATE_MUTE 211
7.30.1.34 GET_AIR_INFO_TELECONTROLLER_MODE_AUTO 212
7.30.1.35 GET_AIR_INFO_TELECONTROLLER_MODE_COLD 212
7.30.1.36 GET_AIR_INFO_TELECONTROLLER_MODE_DRY 212

7.30.1.37 GET_AIR_INFO_TELECONTROLLER_MODE_HOT 212
7.30.1.38 GET_AIR_INFO_TELECONTROLLER_MODE_WIND 212
7.30.1.39 GET_AIR_INFO_TEMPERATURE_OVERFLOW 212
7.30.1.40 GET_AIR_INFO_TEMPERATURE_UNDERFLOW 212
7.30.1.41 GET_AIR_INFO_TEMPERATURE_UNKNOWN 212
7.30.1.42 HIGH_CHAR 212
7.30.1.43 LOW_CHAR 212
7.30.1.44 PACKAGE_MAX 212
7.30.1.45 RECEIVE_CHK_ERROR 212
7.30.1.46 RECEIVE_ERROR 213
7.30.1.47 RESET_AIR_CID_SEND 213
7.30.1.48 RETURN_CHK_ERROR 213
7.30.1.49 RETURN_DATA_INVALID_ERROR 213
7.30.1.50 RTN_RECEIVE_CHK_ERROR 213
7.30.1.51 RTN_RECEIVE_CMD_INVALID 213
7.30.1.52 RTN_RECEIVE_CMD_RIGHT 213
7.30.1.53 RTN_SEND 213
7.30.1.54 SEND_ERROR 213
7.30.1.55 SET_AIR_ARG_CID_SEND 213
7.30.1.56 SET_AIR_ARG_CONFIG_OPT_ALL 213
7.30.1.57 SET_AIR_ARG_CONFIG_OPT_KEEP 213
7.30.1.58 SET_AIR_ARG_CONFIG_OPT_MODE 214
7.30.1.59 SET_AIR_ARG_CONFIG_OPT_SWITCH 214
7.30.1.60 SET_AIR_ARG_CONFIG_OPT_TEMPERATURE 214
7.30.1.61 SET_AIR_ARG_CONFIG_OPT_WIND_DIRECTION 214
7.30.1.62 SET_AIR_ARG_CONFIG_OPT_WIND_RATE 214
7.30.1.63 SET_AIR_ARG_OFF 214
7.30.1.64 SET_AIR_ARG_ON 214
7.30.1.65 SET_AIR_ARG_TELECONTROLLER_MODE_AUTO 214
7.30.1.66 SET_AIR_ARG_TELECONTROLLER_MODE_COLD 214
7.30.1.67 SET_AIR_ARG_TELECONTROLLER_MODE_DRY 214
7.30.1.68 SET_AIR_ARG_TELECONTROLLER_MODE_HOT 214
7.30.1.69 SET_AIR_ARG_TELECONTROLLER_MODE_WIND 214
7.30.1.70 SET_AIR_ARG_TEMPERATURE 215
7.30.1.71 SET_AIR_ARG_WIND_DIRECTION_FOCUS 215
7.30.1.72 SET_AIR_ARG_WIND_DIRECTION_HANDL 215
7.30.1.73 SET_AIR_ARG_WIND_DIRECTION_WIDE 215
7.30.1.74 SET_AIR_ARG_WIND_RATE_AUTO 215
7.30.1.75 SET_AIR_ARG_WIND_RATE_HIGH 215
7.30.1.76 SET_AIR_ARG_WIND_RATE_LOW 215

7.30.1.77 SET_AIR_ARG_WIND_RATE_MIDDLE 215
7.30.1.78 SET_AIR_ARG_WIND_RATE_MOST 215
7.30.1.79 SET_AIR_ARG_WIND_RATE_MUTE 215
7.30.1.80 SOI_RECEIVE 215
7.30.1.81 SOI_SEND 215
7.30.1.82 UNKNOW_ERROR 216
7.30.2 Function Documentation 216
7.30.2.1 calculate_sum_check 216
7.30.2.2 panasonnic_send_package 216
7.30.3 Variable Documentation 216
7.30.3.1 air_command_table 217
7.31 src/device/airCondition/york/york.c File Reference 217
7.31.1 Enumeration Type Documentation 218
7.31.1.1 york_air_conditioner_object 218
7.31.2 Function Documentation 219
7.31.2.1 get_air_york_write_args 219
7.31.3 Variable Documentation 219
7.31.3.1 york_air_condition_object 219
7.31.3.2 york_air_condition_read_property 219
7.32 src/device/curtain/aoke/aoke.c File Reference 219
7.32.1 Macro Definition Documentation 221
7.32.1.1 RS485_CURTAIN_AO_KE_ADDRING 221
7.32.1.2 RS485_CURTAIN_AO_KE_COMMAND_ADDRING 221
7.32.1.3 RS485_CURTAIN_AO_KE_COMMAND_CONTROL 221
7.32.1.4 RS485_CURTAIN_AO_KE_COMMAND_DELETE 221
7.32.1.5 RS485_CURTAIN_AO_KE_COMMAND_GETTING 221
7.32.1.6 RS485_CURTAIN_AO_KE_COMMAND_POINT 221
7.32.1.7 RS485_CURTAIN_AO_KE_COMMAND_POSTION 221
7.32.1.8 RS485_CURTAIN_AO_KE_COMMAND_SETTING 221
7.32.1.9 RS485_CURTAIN_AO_KE_CONTROL_DELETE_ADDR 221
7.32.1.10 RS485_CURTAIN_AO_KE_CONTROL_DOWN 221
7.32.1.11 RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_1 221
7.32.1.12 RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_2 221
7.32.1.13 RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_3 222
7.32.1.14 RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_4 222
7.32.1.15 RS485_CURTAIN_AO_KE_CONTROL_SET_ADDR 222
7.32.1.16 RS485_CURTAIN_AO_KE_CONTROL_STOP 222
7.32.1.17 RS485_CURTAIN_AO_KE_CONTROL_UP 222
7.32.1.18 RS485_CURTAIN_AO_KE_DELETE 222
7.32.1.19 RS485_CURTAIN_AO_KE_POINT_DELETE 222

7.32.1.20 RS485_CURTAIN_AO_KE_POINT_DOWN . . . 222
7.32.1.21 RS485_CURTAIN_AO_KE_POINT_MIDDLE . . . 222
7.32.1.22 RS485_CURTAIN_AO_KE_POINT_SAVE . . . 222
7.32.1.23 RS485_CURTAIN_AO_KE_POINT_UP . . . 222
7.32.1.24 RS485_CURTAIN_AO_KE_SI . . . 222
7.32.1.25 RS485_CURTIAN_AO_KE_GETTING_STATUS . . . 223
7.32.1.26 RS485_CURTIAN_AO_KE_POSTION . . . 223
7.32.1.27 RS485_CURTIAN_AO_KE_SETTING_HANDLE . . . 223
7.32.2 Function Documentation . . . 223
7.32.2.1 get_check . . . 223
7.33 src/device/curtain/doya/doya.c File Reference . . . 223
7.33.1 Macro Definition Documentation . . . 225
7.33.1.1 CURTAIN_COMMAND . . . 225
7.33.1.2 CURTAIN_COMMAND_CLOSE . . . 225
7.33.1.3 CURTAIN_COMMAND_DELETE . . . 225
7.33.1.4 CURTAIN_COMMAND_OPEN . . . 225
7.33.1.5 CURTAIN_COMMAND_PERCENT . . . 225
7.33.1.6 CURTAIN_COMMAND_REFACTORY . . . 226
7.33.1.7 CURTAIN_COMMAND_STOP . . . 226
7.33.1.8 CURTAIN_READ . . . 226
7.33.1.9 CURTAIN_READ_WRITE_ADDR . . . 226
7.33.1.10 CURTAIN_READ_WRITE_ADDR_HIGH . . . 226
7.33.1.11 CURTAIN_READ_WRITE_ADDR_LOW . . . 226
7.33.1.12 CURTAIN_READ_WRITE_DIRECTION . . . 226
7.33.1.13 CURTAIN_READ_WRITE_HANDLE . . . 226
7.33.1.14 CURTAIN_READ_WRITE_PERCENT . . . 226
7.33.1.15 CURTAIN_READ_WRITE_SWITCH_ACTIVE . . . 226
7.33.1.16 CURTAIN_READ_WRITE_SWITCH_PASSIVE . . . 226
7.33.1.17 CURTAIN_READ_WRITE_VERSION . . . 226
7.33.1.18 CURTAIN_WRITE . . . 227
7.33.1.19 HANDLE_DISABLE . . . 227
7.33.1.20 HANDLE_ENABLE . . . 227
7.33.1.21 MOTOR_NEGATIVE . . . 227
7.33.1.22 MOTOR_POSITIVE . . . 227
7.33.1.23 SWITCH_ACTIVE_DOUBLE_LINE . . . 227
7.33.1.24 SWITCH_ACTIVE_SINGLE_LINE . . . 227
7.33.1.25 SWITCH_PASSIVE_DOUBLE_NO_REBOUND . . . 227
7.33.1.26 SWITCH_PASSIVE_DOUBLE_REBOUND . . . 227
7.33.1.27 SWITCH_PASSIVE_ELECTRONIC_DC246 . . . 227
7.33.1.28 SWITCH_PASSIVE_SINGLE_CYCLE . . . 227

7.33.2 Enumeration Type Documentation 227
7.33.2.1 command 227
7.33.3 Function Documentation 228
7.33.3.1 close_curtain_no_reply 228
7.33.3.2 delete_track_curtain_no_reply 228
7.33.3.3 doya_send_package 229
7.33.3.4 modbus_crc16 229
7.33.3.5 open_curtain_no_reply 230
7.33.3.6 percent_curtain_no_reply 231
7.33.3.7 read_percent_curtain_no_reply 231
7.33.3.8 read_version_curtain_no_reply 231
7.33.3.9 receive_package 232
7.33.3.10 refactory_curtain_no_reply 232
7.33.3.11 rs485_read 232
7.33.3.12 rs485_send 233
7.33.3.13 send_package 234
7.33.3.14 set_addr_curtain_no_reply 236
7.33.3.15 set_direction_curtain_no_reply 237
7.33.3.16 set_handle_enable_curtain_no_reply 237
7.33.3.17 set_switch_active_curtain_no_reply 237
7.33.3.18 set_switch_passive_curtain_no_reply 238
7.33.3.19 stop_curtain_no_reply 238
7.33.4 Variable Documentation 238
7.33.4.1 crc_high 238
7.33.4.2 crc_low 239
7.34 src/device/freshAir/loreley/loreley.c File Reference 239
7.34.1 Macro Definition Documentation 241
7.34.1.1 ADR_BROADCAST 241
7.34.1.2 ADR_DEFAULT 241
7.34.1.3 EOI 241
7.34.1.4 NEW_TREND_PACKAGE_MAX 241
7.34.1.5 RS485_NEW_TREND_GET_ADDR_CID 241
7.34.1.6 RS485_NEW_TREND_GET_INFO_CID 241
7.34.1.7 RS485_NEW_TREND_MODE_AUTOING 241
7.34.1.8 RS485_NEW_TREND_MODE_IN_CRC 241
7.34.1.9 RS485_NEW_TREND_MODE_KILLING 241
7.34.1.10 RS485_NEW_TREND_MODE_OUT_CRC 241
7.34.1.11 RS485_NEW_TREND_MODE_WAITING 241
7.34.1.12 RS485_NEW_TREND_RESTART_CID 241
7.34.1.13 RS485_NEW_TREND_RUN_STATUS_OFF 242

7.34.1.14 RS485_NEW_TREND_RUN_STATUS_ON 242
7.34.1.15 RS485_NEW_TREND_SET_ADDR_CID 242
7.34.1.16 RS485_NEW_TREND_SET_ARG_CID 242
7.34.1.17 RS485_NEW_TREND_STATUS_ERROR 242
7.34.1.18 RS485_NEW_TREND_STATUS_NORMAL 242
7.34.1.19 RTN_RECEIVE_CHK_ERROR 242
7.34.1.20 RTN_RECEIVE_CMD_INVALID 242
7.34.1.21 RTN_RECEIVE_CMD_RIGHT 242
7.34.1.22 RTN_SEND 242
7.34.1.23 SOI_RECEIVE 242
7.34.1.24 SOI_SEND 242
7.34.2 Function Documentation 243
7.34.2.1 calculate_sum_check 243
7.34.2.2 loreley_send_package 243
7.34.3 Variable Documentation 243
7.34.3.1 rs485_set_new_trend_table 243
7.35 src/enumtxt.c File Reference 244
7.36 src/item_config.c File Reference 244
7.36.1 Macro Definition Documentation 245
7.36.1.1 PANNO_S_ITEM_ARMANI 245
7.36.1.2 PANNO_S_ITEM_DEFAULT 245
7.36.1.3 PANNO_S_ITEM_SHAOCHENGGUOJI 245
7.36.1.4 PANNO_S_ITEM_WENRUDE 246
7.36.2 Function Documentation 246
7.36.2.1 _mount_device 246
7.37 src/main.c File Reference 246
7.37.1 Detailed Description 247
7.37.2 Macro Definition Documentation 247
7.37.2.1 LOCKFILE 247
7.37.2.2 LOCKMODE 247
7.37.3 Function Documentation 247
7.37.3.1 already_running 247
7.37.3.2 daemonize 248
7.37.3.3 lockfile 248
7.37.3.4 main 248
7.37.3.5 signal_handle_pthread 248
7.37.4 Variable Documentation 248
7.37.4.1 mask 248
7.38 src/object.c File Reference 249
7.38.1 Macro Definition Documentation 250

7.38.1.1 RS485_OBJECT_MAX_NUMBERS 250
7.38.1.2 RS485_WORK_QUEUE_DEPTH 250
7.38.2 Function Documentation 250
7.38.2.1 check_object_is_support 250
7.38.3 Variable Documentation 251
7.38.3.1 glb_object_manage 251
7.38.3.2 glb_work_queue 251
7.39 src/protocol/bacnet/bacnet.c File Reference 251
7.39.1 Function Documentation 252
7.39.1.1 york_air_condition_handle 252
7.40 src/protocol/bacnet/device-client.c File Reference 252
7.40.1 Detailed Description 254
7.40.2 Function Documentation 254
7.40.2.1 Device_Application_Software_Version 254
7.40.2.2 Device_Count 254
7.40.2.3 Device_Database_Revision 254
7.40.2.4 Device_Description 254
7.40.2.5 Device_Firmware_Revision 254
7.40.2.6 Device_Inc_Database_Revision 254
7.40.2.7 Device_Index_To_Instance 255
7.40.2.8 Device_Init 255
7.40.2.9 Device_Location 255
7.40.2.10 Device_Model_Name 255
7.40.2.11 Device_Object_Instance_Number 255
7.40.2.12 Device_Object_List_Count 256
7.40.2.13 Device_Object_List_Identifier 256
7.40.2.14 Device_Object_Name 257
7.40.2.15 Device_Object_Name_Copy 257
7.40.2.16 Device_Objects_Find_Functions 258
7.40.2.17 Device_Protocol_Revision 258
7.40.2.18 Device_Protocol_Version 258
7.40.2.19 Device_Read_Property 259
7.40.2.20 Device_Read_Property_Local 259
7.40.2.21 Device_Segmentation_Supported 260
7.40.2.22 Device_Set_Application_Software_Version 260
7.40.2.23 Device_Set_Database_Revision 260
7.40.2.24 Device_Set_Description 261
7.40.2.25 Device_Set_Location 261
7.40.2.26 Device_Set_Model_Name 261
7.40.2.27 Device_Set_Object_Instance_Number 261

7.40.2.28 Device_Set_Object_Name 261
7.40.2.29 Device_Set_System_Status 262
7.40.2.30 Device_Set_Vendor_Identifier 262
7.40.2.31 Device_System_Status 262
7.40.2.32 Device_Valid_Object_Id 262
7.40.2.33 Device_Valid_Object_Instance_Number 262
7.40.2.34 Device_Valid_Object_Name 262
7.40.2.35 Device_Vendor_Identifier 263
7.40.2.36 Device_Vendor_Name 263
7.40.3 Variable Documentation 263
7.40.3.1 Application_Software_Version 263
7.40.3.2 Database_Revision 263
7.40.3.3 Description 263
7.40.3.4 Location 263
7.40.3.5 Model_Name 264
7.40.3.6 My_Object_Name 264
7.40.3.7 Object_Instance_Number 264
7.40.3.8 Object_Table 264
7.40.3.9 System_Status 264
7.40.3.10 Vendor_Identifier 264
7.40.3.11 Vendor_Name 264
7.41 src/protocol/bacnet/handle_property.c File Reference 265
7.41.1 Macro Definition Documentation 266
7.41.1.1 MAX_PROPERTY_VALUES 266
7.41.2 Function Documentation 266
7.41.2.1 enno_dlenv_init 266
7.41.2.2 Init_Service_Handlers 266
7.41.2.3 My_Read_Property_Ack_Handler 267
7.41.2.4 My_Read_Property_Multiple_Ack_Handler 268
7.41.2.5 MyAbortHandler 268
7.41.2.6 MyErrorHandler 268
7.41.2.7 MyRejectHandler 268
7.41.2.8 MyWritePropertySimpleAckHandler 268
7.41.3 Variable Documentation 269
7.41.3.1 glb_config_bacnet_object_instance 269
7.42 src/protocol/bacnet/read_property.c File Reference 269
7.42.1 Function Documentation 270
7.42.1.1 bacnet_read_property 270
7.42.1.2 memcpy_read_args 270
7.42.1.3 My_Read_Property_Ack_Handler 271

7.42.1.4 My_Read_Property_Multiple_Ack_Handler 271
7.42.2 Variable Documentation 271
7.42.2.1 read_access_data 271
7.42.2.2 read_access_data_property 271
7.42.2.3 Request_Invoke_ID 271
7.42.2.4 Target_Address 271
7.43 src/protocol/bacnet/write_property.c File Reference 272
7.43.1 Macro Definition Documentation 273
7.43.1.1 MAX_PROPERTY_VALUES 273
7.43.1.2 RETRANSMISSION_TIMES 273
7.43.2 Function Documentation 273
7.43.2.1 bacnet_write_property 273
7.43.2.2 MyWritePropertySimpleAckHandler 273
7.43.3 Variable Documentation 273
7.43.3.1 Error_Detected 273
7.43.3.2 Request_Invoke_ID 273
7.43.3.3 Rx_Buf 273
7.43.3.4 Target_Address 274
7.43.3.5 Target_Object_Property_Value 274
7.44 src/protocol/general/general.c File Reference 274
7.44.1 Macro Definition Documentation 275
7.44.1.1 BUS_MAX_RETRANSMISSION 275
7.45 src/protocol/general/rs485.c File Reference 275
7.45.1 Macro Definition Documentation 276
7.45.1.1 _POSIX_SOURCE 276
7.45.1.2 RS485_DEBUG 276
7.45.1.3 RS485MOD 276
7.45.2 Function Documentation 276
7.45.2.1 rs485_get_baud_rate 276
7.45.2.2 rs485_interface 276
7.45.3 Variable Documentation 276
7.45.3.1 RS485_Baud 276
7.45.3.2 RS485_Handle 277
7.45.3.3 RS485_oldserial 277
7.45.3.4 RS485_oldtio 277
7.45.3.5 RS485_Port_Name 277
7.45.3.6 RS485_SpecBaud 277
7.46 src/protocol/modbus/modbus.c File Reference 277
7.46.1 Macro Definition Documentation 278
7.46.1.1 RS485_MODBUS_MTU 278

7.47 src/read_config.c File Reference 278
7.47.1 Variable Documentation 279
7.47.1.1 glb_config_adapter_message_queue_depth 279
7.47.1.2 glb_config_bacnet_object_instance 279
7.47.1.3 glb_config_bacnet_work_queue_depth 279
7.47.1.4 glb_config_client_max_numbers 279
7.47.1.5 glb_config_general_work_package_mtu 279
7.47.1.6 glb_config_general_work_queue_depth 279
7.47.1.7 glb_config_modbus_work_queue_depth 279
7.48 src/service.c File Reference 279
7.48.1 Detailed Description 281
7.48.2 Macro Definition Documentation 281
7.48.2.1 NALLOC 281
7.48.2.2 RECEIVE_BUFFER_LENGTH 281
7.48.2.3 RS485_UNIX_DOMAIN_PATH 281
7.48.3 Function Documentation 281
7.48.3.1 adapter_thread_function 281
7.48.3.2 client_add 282
7.48.3.3 client_alloc 283
7.48.3.4 client_del 283
7.48.3.5 process_client_request 283
7.48.3.6 rs485_receive_from_client 284
7.48.3.7 rs485_service_listen 284
7.48.3.8 serv_accept 284
7.48.3.9 timer_task_thread_function 285
7.48.4 Variable Documentation 285
7.48.4.1 adapter_message_queue 285
7.48.4.2 adapter_thread 285
7.48.4.3 addr 285
7.48.4.4 client 285
7.48.4.5 client_size 285
7.48.4.6 glb_config_adapter_message_queue_depth 285
7.48.4.7 glb_config_client_max_numbers 285
7.48.4.8 receive_buffer 285
7.48.4.9 rs485_thread_pool 285
7.48.4.10 socket_fd 286
7.48.4.11 socket_len 286
7.48.4.12 timer_task_thread 286
7.48.4.13 unix_domain_path 286
7.49 src/support.c File Reference 286

7.49.1 Variable Documentation 287
7.49.1.1 air_condition_daikin_dta116a621 287
7.49.1.2 air_condition_panasonnic 287
7.49.1.3 curtain_aoke 287
7.49.1.4 curtain_doya 288
7.49.1.5 fresh_air_loreley 288
7.50 src/syslog/log.c File Reference 288
7.50.1 Detailed Description 288
7.51 src/timer_task.c File Reference 289
7.51.1 Detailed Description 290
7.51.2 Macro Definition Documentation 290
7.51.2.1 SYSTEM_TIMER_TICK_SECOND 290
7.51.3 Function Documentation 290
7.51.3.1 device_timer_task_handle_curtain_aoke_init 290
7.51.3.2 device_timer_task_handle_curtain_doya_init 290
7.51.3.3 device_timer_task_handle_curtain_init 291
7.51.3.4 timer_task_init 292
7.51.3.5 timer_task_thread_function 292
7.51.4 Variable Documentation 292
7.51.4.1 timer_task_list 292
7.51.4.2 timer_task_list_iterator 292
7.51.4.3 timer_task_lock 292
7.51.4.4 timer_task_thread_status 293

Index **294**

Chapter 1

RS485 SERVER API Documentation

This documents the RS485 API, Modbus/BACnet/General, and sample applications.

- The high-level handler interface can be found in the Modules tab.
- Specifics for each file can be found in the Files tab.
- A full list of all functions is provided in the index of the Files->Globals subtab.

While all the central files are included in the file list, not all important functions have been given the javadoc treatment, nor have Modules (chapters) been created yet for all groupings. If you are doing work in an under- documented area, please add the javadoc comments at least to the API calls, and consider adding doxygen's module grouping for your area of interest.

See doc/README.doxygen for notes on building and extending this document.
In particular, if you have graphviz installed, you can enhance this documentation by turning on the function call graphs feature.

```
Copyright: www.enno.com

Author: chuanjiang.wang
E-mail:  chuanjiang.wang@enno.com
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Management . . . . . . ??
  Adapter management . . . . . . ??
  Device management . . . . . . ??
  String management . . . . . . ??
  Item management . . . . . . ??
  Object management . . . . . . ??
  Service management . . . . . . ??
  Device register management . . . . . . ??
  Timer management . . . . . . ??
Protocol . . . . . . ??
  BACnet . . . . . . ??
    BACnet interface . . . . . . ??
    Handle_property . . . . . . ??
  Modbus . . . . . . ??
    Modbus RS485 . . . . . . ??
  General . . . . . . ??
    General interface . . . . . . ??
    General RS485 . . . . . . ??
Device . . . . . . ??
  AirConditon . . . . . . ??
    DaiKin DTA116A621 . . . . . . ??
    Panasonnic GuangZhou . . . . . . ??
    YORK GuangZhou KeLong . . . . . . ??
  Curtain . . . . . . ??
    Aoke GuangZhou . . . . . . ??
    Dooya HangZhou . . . . . . ??
  FreshAir . . . . . . ??
    Loreley ShenZhen . . . . . . ??

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

adapter_t
Define the adapter struct . . . . ??
air_condition_profile_t
The air conditon profile . . . . ??
bacnet
Bacnet bacnet interface struct . . . . ??
bacnet_read_args_t
Bacnet read property struct . . . . ??
bacnet_write_args_t
Bacnet write arg struct . . . . ??
client_t . . . . ??
commonBacObj_s . . . . ??
create_object_return_t
Message create a rs485 object return . . . . ??
create_object_t
Message create a rs485 object . . . . ??
curtain_profile_t
The curtain profile . . . . ??
delete_object_return_t
Message delete a rs485 object return . . . . ??
delete_object_t
Message delete a rs485 object . . . . ??
device_management
Device define the device management struct . . . . ??
device_profile
Device_profile device process method . . . . ??
devObj_s . . . . ??
fresh_air_profile_t
The fresh profile . . . . ??
message_service_t
Define the receive the message type . . . . ??
modbus_port_handle_t
The modbus port interface . . . . ??
mount_devcie_to_object_t
Message mount a device to rs485 object . . . . ??
mount_device_to_object_return_t
Message mount a device to rs485 object return . . . . ??

mstp_port_handle
Mstp_port_handle general protocol(user defined) . . . . ??
object_functions . . . . ??
object_management
Object_management define the object management struct . . . . ??
package . . . . ??
read_device_return_t
Message read value from device return . . . . ??
read_device_t
Message read value from device . . . . ??
rs485_curtain_ao_ke_send_package_t . . . . ??
rs485_device_profile
Rs485 device profile . . . . ??
rs485_port_t
The rs485 port physical . . . . ??
thread_pool_t
Define the thread pool struct . . . . ??
timer_task_t
Timer task struct . . . . ??
unmount_device_from_object_return_t
Message unmount a device from rs485 ojbect return . . . . ??
unmount_device_from_object_t
Message unmount a device form rs485 object . . . . ??
write_device_return_t
Message write value to device return . . . . ??
write_device_t
Message write value to device . . . . ??

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/adapter.h . . . . ??
include/device.h . . . . ??
include/enum.h . . . . ??
include/enumtxt.h . . . . ??
include/item_config.h . . . . ??
include/object.h . . . . ??
include/read_config.h . . . . ??
include/rs485.h . . . . ??
include/service.h . . . . ??
include/support.h . . . . ??
include/timer_task.h . . . . ??
include/device/airCondition/daikin/DTA116A621.h . . . . ??
include/device/airCondition/panasonnic/panasonnic.h . . . . ??
include/device/airCondition/york/york.h . . . . ??
include/device/curtain/aoke/aoke.h . . . . ??
include/device/curtain/doya/doya.h . . . . ??
include/device/freshAir/loreley/loreley.h . . . . ??
include/protocol/bacnet/bacnet.h . . . . ??
include/protocol/bacnet/device_client.h . . . . ??
include/protocol/bacnet/handle_property.h . . . . ??
include/protocol/bacnet/read_property.h . . . . ??
include/protocol/bacnet/write_property.h . . . . ??
include/protocol/general/general.h . . . . ??
include/protocol/general/rs485.h . . . . ??
include/protocol/modbus/modbus.h . . . . ??
include/syslog/log.h . . . . ??
src/adapter.c . . . . ??
src/device.c . . . . ??
src/enumtxt.c . . . . ??
src/item_config.c . . . . ??
src/main.c . . . . ??
src/object.c . . . . ??
src/read_config.c . . . . ??
src/service.c . . . . ??
src/support.c . . . . ??
src/timer_task.c . . . . ??
src/device/airCondition/daikin/DTA116A621.c . . . . ??
src/device/airCondition/panasonnic/panasonnic.c . . . . ??

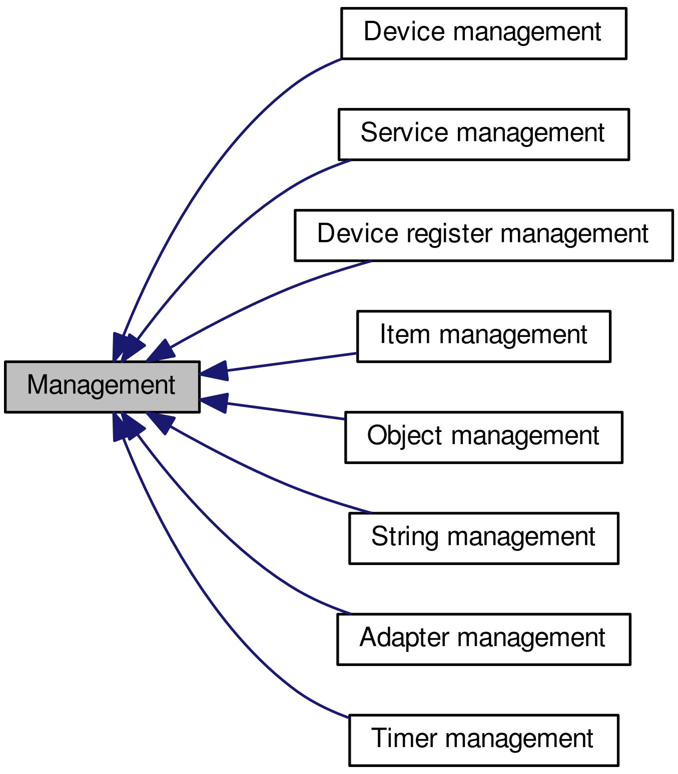
src/device/airCondition/york/york.c . . . . ??
src/device/curtain/aoke/aoke.c . . . . ??
src/device/curtain/doya/doya.c . . . . ??
src/device/freshAir/loreley/loreley.c . . . . ??
src/protocol/bacnet/bacnet.c . . . . ??
src/protocol/bacnet/device-client.c . . . . ??
src/protocol/bacnet/handle_property.c . . . . ??
src/protocol/bacnet/read_property.c . . . . ??
src/protocol/bacnet/write_property.c . . . . ??
src/protocol/general/general.c . . . . ??
src/protocol/general/rs485.c . . . . ??
src/protocol/modbus/modbus.c . . . . ??
src/syslog/log.c . . . . ??

Chapter 5

Module Documentation

5.1 Management

Collaboration diagram for Management:



Modules

- Adapter management
- Device management

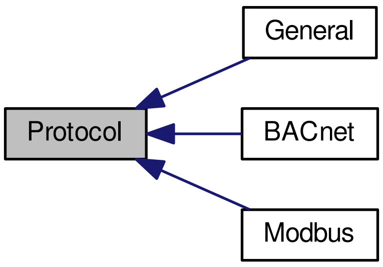
- String management
- Item management
- Object management
- Service management
- Device register management
- Timer management

5.1.1 Detailed Description

The RS485 service management

5.2 Protocol

Collaboration diagram for Protocol:



Modules

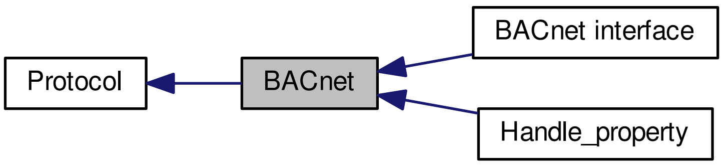
- BACnet
- Modbus
- General

5.2.1 Detailed Description

The rs485 support protocl module

5.3 BACnet

Collaboration diagram for BACnet:



Modules

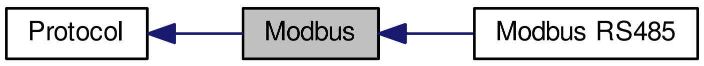
- BACnet interface
- Handle_property

5.3.1 Detailed Description

The BACnet protocol

5.4 Modbus

Collaboration diagram for Modbus:



Modules

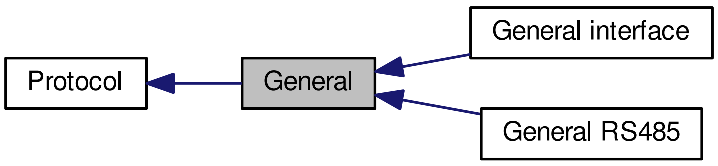
- Modbus RS485

5.4.1 Detailed Description

The Modbus protocl

5.5 General

Collaboration diagram for General:



Modules

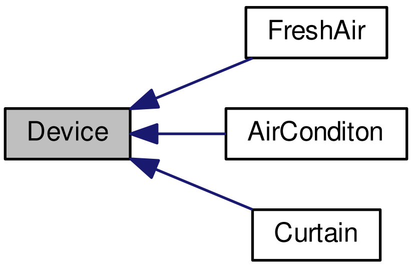
- General interface
- General RS485

5.5.1 Detailed Description

The user defined protocl

5.6 Device

Collaboration diagram for Device:



Modules

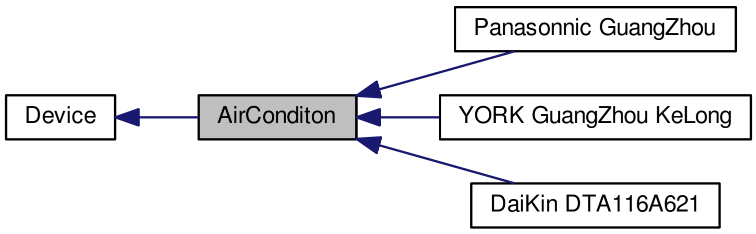
- AirConditon
- Curtain
- FreshAir

5.6.1 Detailed Description

The rs485 support device module

5.7 AirConditon

Collaboration diagram for AirConditon:



Modules

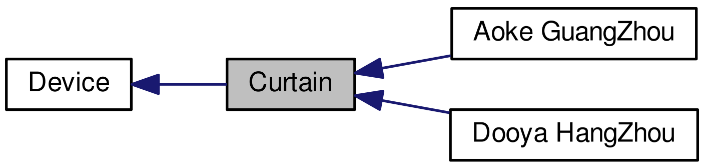
- DaiKin DTA116A621
- Panasonnic GuangZhou
- YORK GuangZhou KeLong

5.7.1 Detailed Description

The air conditon device

5.8 Curtain

Collaboration diagram for Curtain:



Modules

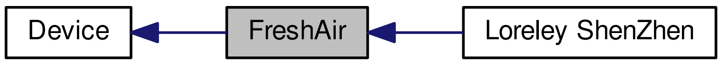
- Aoke GuangZhou
- Dooya HangZhou

5.8.1 Detailed Description

The curtain device

5.9 FreshAir

Collaboration diagram for FreshAir:



Modules

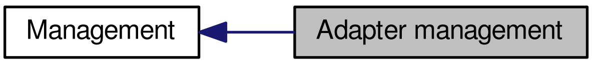
- Loreley ShenZhen

5.9.1 Detailed Description

The fresh air device

5.10 Adapter management

Collaboration diagram for Adapter management:



Data Structures

- struct rs485_port_t

  The rs485 port physical.
- struct create_object_t

  message create a rs485 object
- struct create_object_return_t

  message create a rs485 object return
- struct delete_object_t

  message delete a rs485 object
- struct delete_object_return_t

  message delete a rs485 object return
- struct mount_devcie_to_object_t

  message mount a device to rs485 object
- struct mount_device_to_object_return_t

  message mount a device to rs485 object return
- struct unmount_device_from_object_t

  message unmount a device form rs485 object
- struct unmount_device_from_object_return_t

  message unmount a device from rs485 ojbect return
- struct write_device_t

  message write value to device
- struct write_device_return_t

  message write value to device return
- struct read_device_t

  message read value from device
- struct air_condition_profile_t

  The air conditon profile.
- struct curtain_profile_t

  The curtain profile.
- struct fresh_air_profile_t

  The fresh profile.
- union rs485_device_profile

  rs485 device profile
- struct read_device_return_t

  message read value from device return
- union message_service_t

  define the receive the message type
- struct adapter_t

  define the adapter struct

Functions

- static int adapter_thread_init (void)

    adapter_thread_init initialze the adapter thread , and mesesage queue initial.
- static int process_write_value_service (const adapter_t ∗adapter)

    process_write_value_service process the client write value to device service
- static int process_read_value_service (adapter_t ∗adapter)

    process_read_value_service process the client read value from device service

5.10.1 Detailed Description

Functions to rs485 device create , delete , management.

5.10.2 Function Documentation

5.10.2.1 static int adapter_thread_init (void) [static]

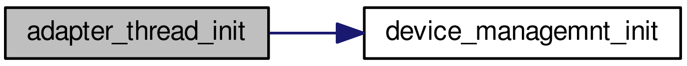
adapter_thread_init initialze the adapter thread , and mesesage queue initial.

Returns

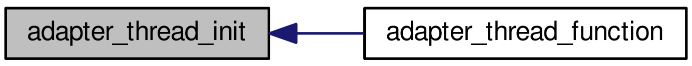
0 is initialize success, and others is fail.

Definition at line 65 of file adapter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.2 static int process_read_value_service (adapter_t ∗ *adapter*) [static]

process_read_value_service process the client read value from device service

Parameters

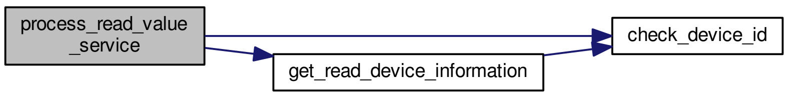
adapter	: The adapter struct information

Returns

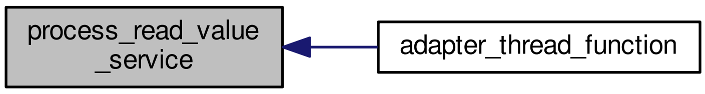
0 is success, and others is fail.

Definition at line 285 of file adapter.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.10.2.3 static int process_write_value_service (const adapter_t ∗ *adapter*) [static]

process_write_value_service process the client write value to device service

Parameters

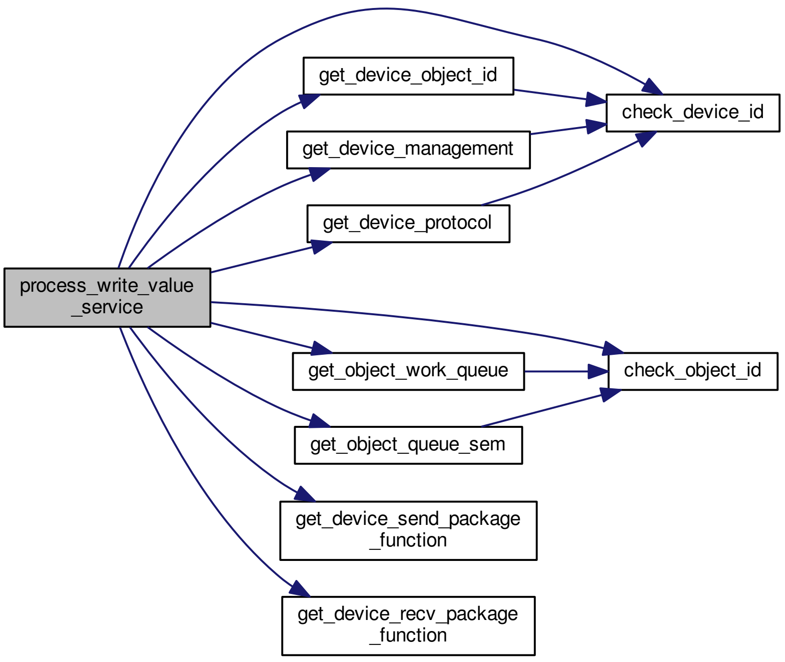
in	*adapter*	: The adapter struct information

Returns

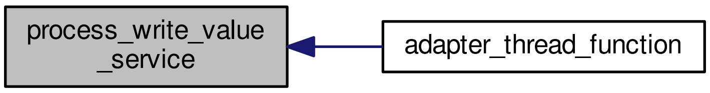
0 is success, and others is fail.

Definition at line 94 of file adapter.c.

Here is the call graph for this function:

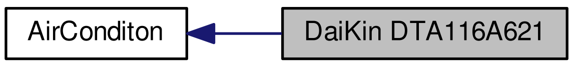


Here is the caller graph for this function:



5.11 DaiKin DTA116A621

Collaboration diagram for DaiKin DTA116A621:



Functions

- int daikin_dta116a621_set_temperature (volatile void ∗arg)

  daikin_dta116a621_set_temperature set daikin air condition temperature send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_mode (volatile void ∗arg)

  daikin_dta116a621_set_mode set daikin air conditon mode send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_swing (volatile void ∗arg)

  daikin_dta116a621_set_swing set daikin air conditon swing send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_fan (volatile void ∗arg)

  daikin_dta116a621_set_fan set daikin air conditon fan send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_switch (volatile void ∗arg)

  daikin_dta116a621_set_switch set daikin air conditon switch send package to "modbus_port_handle_t"
- int daikin_dta116a621_get_device_info_send (volatile void ∗arg)

  daikin_dta116a621_get_device_info_send set daikin air conditon device information send package to "modbus_↩ port_handle_t"
- int daikin_dta116a621_get_device_info_handle (volatile void ∗arg)

  daikin_dta116a621_get_device_info_handle process daikin air conditon get device information send package to "modbus_port_handle_t"

5.11.1 Detailed Description

Functions to DaiKin DTA116A621 interface.

5.11.2 Function Documentation

5.11.2.1 int daikin_dta116a621_get_device_info_handle (volatile void ∗ *arg*)

daikin_dta116a621_get_device_info_handle process daikin air conditon get device information send package to "modbus_port_handle_t"

Parameters

`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

0 is success, others is fail.

Definition at line 507 of file DTA116A621.c.

5.11.2.2 int daikin_dta116a621_get_device_info_send (volatile void ∗ *arg*)

daikin_dta116a621_get_device_info_send set daikin air conditon device information send package to "modbus_↩
port_handle_t"

Parameters

`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

Return the set device information send package length, if have a error , return a negative value.

Definition at line 495 of file DTA116A621.c.

5.11.2.3 int daikin_dta116a621_set_fan (volatile void * *arg*)

daikin_dta116a621_set_fan set daikin air conditon fan send package to "modbus_port_handle_t"

Parameters

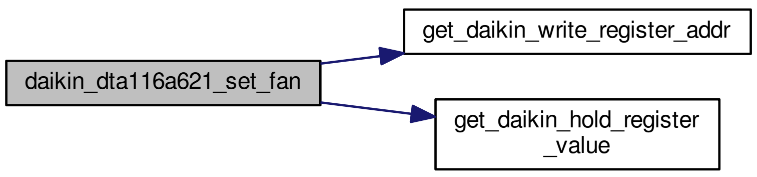
`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

Return the set fan send package length, if have a error , return a negative value.

Definition at line 428 of file DTA116A621.c.

Here is the call graph for this function:



5.11.2.4 int daikin_dta116a621_set_mode (volatile void * *arg*)

daikin_dta116a621_set_mode set daikin air conditon mode send package to "modbus_port_handle_t"

Parameters

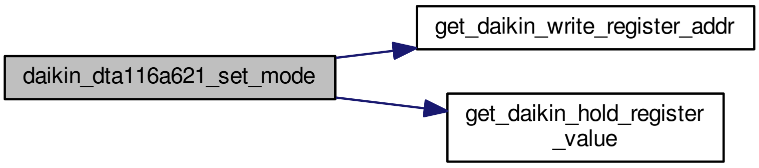
`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

Return the set mode send package length, if have a error , return a negative value.

Definition at line 362 of file DTA116A621.c.

Here is the call graph for this function:



5.11.2.5 int daikin_dta116a621_set_swing (volatile void ∗ *arg*)

daikin_dta116a621_set_swing set daikin air conditon swing send package to "modbus_port_handle_t"

Parameters

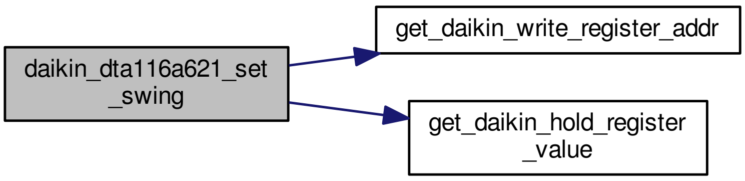
`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

Return the set swing send package length, if have a error , return a negative value.

Definition at line 395 of file DTA116A621.c.

Here is the call graph for this function:



5.11.2.6 int daikin_dta116a621_set_switch (volatile void ∗ *arg*)

daikin_dta116a621_set_switch set daikin air conditon switch send package to "modbus_port_handle_t"

Parameters

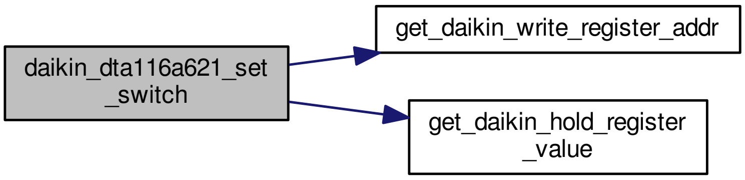
`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

Return the set switch send package length, if have a error , return a negative value.

Definition at line 461 of file DTA116A621.c.

Here is the call graph for this function:



5.11.2.7 int daikin_dta116a621_set_temperature (volatile void ∗ *arg*)

daikin_dta116a621_set_temperature set daikin air condition temperature send package to "modbus_port_handle_t"

Parameters

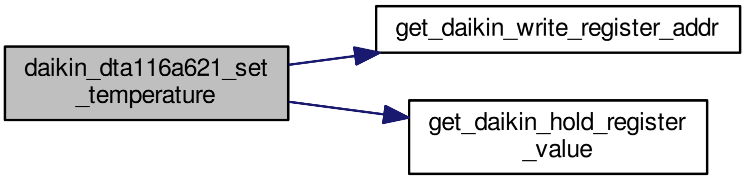
`in,out`	*arg*	: The struct (modbus_port_handle_t) pointer

Returns

Return the set temperature send package length, if have a error ,return a negative value.

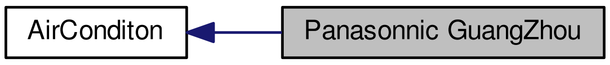
Definition at line 329 of file DTA116A621.c.

Here is the call graph for this function:



5.12 Panasonnic GuangZhou

Collaboration diagram for Panasonnic GuangZhou:



Functions

- int panasonnic_send_package_handle (volatile void ∗arg)

    panasonnic_send_package_handle The panasonnic package a send buffer interface.
- int panasonnic_recv_package_handle (volatile void ∗arg)

    panasonnic_send_package_handle The panasonnic package a receive buffer processs interface.

5.12.1 Detailed Description

Functions to Panasonnic air condition interface.

5.12.2 Function Documentation

5.12.2.1 int panasonnic_recv_package_handle (volatile void ∗ *arg*)

panasonnic_send_package_handle The panasonnic package a receive buffer processs interface.

Parameters

`in`	*arg*	: The struct "mstp_port_handle_t" defined by general.h

Returns

0 is success , others is fail.

Definition at line 1138 of file panasonnic.c.

5.12.2.2 int panasonnic_send_package_handle (volatile void ∗ *arg*)

panasonnic_send_package_handle The panasonnic package a send buffer interface.

Parameters

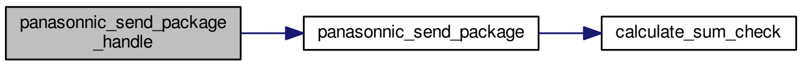
`in,out`	*arg*	: The struct "mstp_port_handle_t" defined by general.h

Returns

Return The send package length, if have a error , return a negative value.

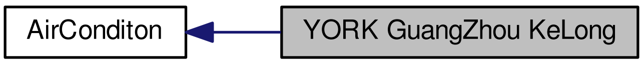
Definition at line 821 of file panasonnic.c.

Here is the call graph for this function:



5.13 YORK GuangZhou KeLong

Collaboration diagram for YORK GuangZhou KeLong:



Functions

- int get_air_york_write_args (bacnet_write_args_t ∗args, unsigned int device_id, int command, int value)

  get_air_york_write_args The york air condition bacnet interface
- int get_air_york_read_args (bacnet_read_args_t ∗args, unsigned int device_id)

  get_air_york_read_args The york air confition bacnet read interface
- int get_air_york_instance (unsigned char mac)

  get_air_york_instance get the youk bacnet instance.

5.13.1 Detailed Description

Functions to york air condition interface.

5.13.2 Function Documentation

5.13.2.1 int get_air_york_instance (unsigned char *mac*)

get_air_york_instance get the youk bacnet instance.

Parameters

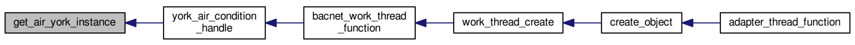
in	*mac*	: The device MAC address

Returns

return the instance, if return negative value is error

Definition at line 301 of file york.c.

Here is the caller graph for this function:



5.13.2.2 int get_air_york_read_args (bacnet_read_args_t ∗ *args*, unsigned int *device_id*)

get_air_york_read_args The york air confition bacnet read interface

Parameters

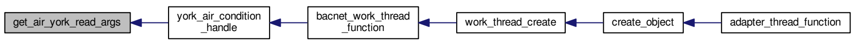
`in,out`	*args*	: The bacnet read struct, so need to full it.
`in`	*device_id*	: The bacnet device id.

Returns

0 is success, and others is fail.

Definition at line 280 of file york.c.

Here is the caller graph for this function:



5.13.2.3 int get_air_york_write_args (bacnet_write_args_t ∗ *args*, unsigned int *device_id*, int *command*, int *value*)

get_air_york_write_args The york air condition bacnet interface

Parameters

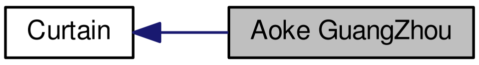
`in,out`	*args*	: The bacnet write struct, so need to full it.
`in`	*device_id*	: The bacnet device id.
`in`	*command*	: The device command, method too.
`in`	*value*	: The value mayto is ununsed.

Returns

0 is success, and others is fail.

5.14 Aoke GuangZhou

Collaboration diagram for Aoke GuangZhou:



Functions

- int aoke_send_package_handle (volatile void ∗arg)

  aoke_send_package_handle aoke curtian package a send buffer
- int aoke_recv_package_handle (volatile void ∗arg)

  aoke_recv_package_handle aoke curtain process the receive package

5.14.1 Detailed Description

Functions to aoke curtain interface.

5.14.2 Function Documentation

5.14.2.1 int aoke_recv_package_handle (volatile void ∗ *arg*)

aoke_recv_package_handle aoke curtain process the receive package

Parameters

`in`	*arg*	: The struct "mstp_port_handle_t" pointer

Returns

0 is success, others is fail.

Definition at line 438 of file aoke.c.

5.14.2.2 int aoke_send_package_handle (volatile void ∗ *arg*)

aoke_send_package_handle aoke curtian package a send buffer

Parameters

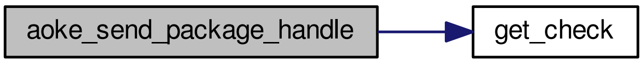
`in,out`	*arg*	: The struct "mstp_port_handle_t" pointer

Returns

The send package length, if have a error return a negative value.

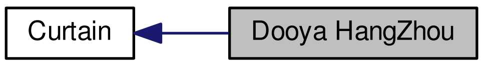
Definition at line 349 of file aoke.c.

Here is the call graph for this function:



5.15 Dooya HangZhou

Collaboration diagram for Dooya HangZhou:



Functions

- int doya_send_package_handle (volatile void ∗arg)

    doya_send_package_handle The dooya curtain package a send buffer
- int doya_recv_package_handle (volatile void ∗arg)

    doya_recv_package_handle The dooya curtain process the receive data.

5.15.1 Detailed Description

Functions to dooya curtain interface.

5.15.2 Function Documentation

5.15.2.1 int doya_recv_package_handle (volatile void ∗ *arg*)

doya_recv_package_handle The dooya curtain process the receive data.

Parameters

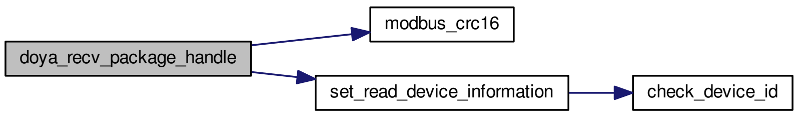
`in`	*arg*	: The struct "mstp_port_handle_t" pointer

Returns

0 is success, others is fail.

Definition at line 886 of file doya.c.

Here is the call graph for this function:



5.15.2.2 int doya_send_package_handle (volatile void ∗ *arg*)

doya_send_package_handle The dooya curtain package a send buffer

Parameters

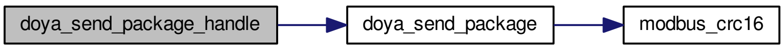
`in,out`	*arg*	: The struct "mstp_port_handle_t" pointer

Returns

The send buffer package length, if have a error, return a negative value.

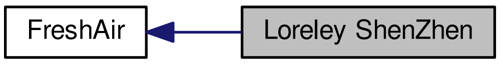
Definition at line 764 of file doya.c.

Here is the call graph for this function:



5.16 Loreley ShenZhen

Collaboration diagram for Loreley ShenZhen:



Functions

- int loreley_send_package_handle (volatile void ∗arg)

    loreley_send_package_handle loreley fresh air package send a buffer
- int loreley_recv_package_handle (volatile void ∗arg)

    loreley_recv_package_handle loreley fresh air process the receive data.

5.16.1 Detailed Description

Functions to loreley fresh air interface.

5.16.2 Function Documentation

5.16.2.1 int loreley_recv_package_handle (volatile void ∗ *arg*)

loreley_recv_package_handle loreley fresh air process the receive data.

Parameters

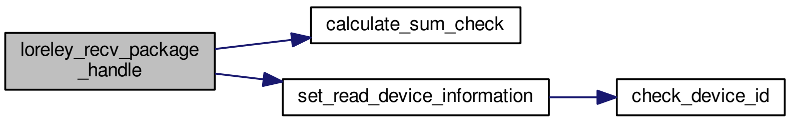
`in`	*arg*	: The struct "mstp_port_handle_t" pointer.

Returns

0 is success, others is fail.

Definition at line 640 of file loreley.c.

Here is the call graph for this function:



5.16.2.2 int loreley_send_package_handle (volatile void ∗ *arg*)

loreley_send_package_handle loreley fresh air package send a buffer

Parameters

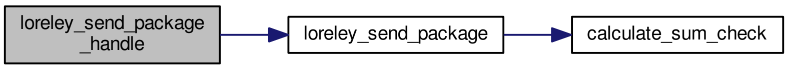
`in,out`	*arg*	: The struct "mstp_port_handle_t" pointer.

Returns

The send buffer package length, if have a error ,return negative value.

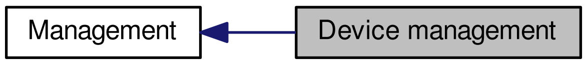
Definition at line 532 of file loreley.c.

Here is the call graph for this function:



5.17 Device management

Collaboration diagram for Device management:



Data Structures

- struct device_management

  device define the device management struct

Typedefs

- typedef struct device_management device_management_t

  device define the device management struct

Functions

- static int find_available_device_id (void)

  find_available_device_id find a available device ID
- int create_device (adapter_t ∗adapter)

  create_device create a rs485 device, mount the device to protocol
- int delete_device (int object_id, int device_id)

  delete_device delete a device form device management table.
- int get_device_name (char ∗out, int out_len, int device_id)

  get_device_name get a device name from device database.
- int get_device_type (int device_id)

  get_device_type get a device type from device database, just like air condition, fresh air.....
- int get_device_protocol (int device_id)

  get_device_protocol get a device protocol from device database, just like BACnet, MODUBS...
- int get_device_addr (unsigned char ∗addr, unsigned int addr_len, int device_id)

  get_device_addr get a rs485 device addr, you maybe have no address for some device.
- timer_task_t ∗ get_device_timer (int device_id)

  get_device_timer get a device timer task.
- struct device_profile ∗ get_device_private (int device_id)

  get_device_private get a device private profile
- int get_device_private_numbers (int device_id)

  get_device_private_numbers
- bool check_device_id (int device_id)

  check_object_id check the object is legal
- int get_device_object_id (int device_id)

  get_device_object_id get the object id by device id
- int get_device_factory_name (int device_id)

get_device_factory_name Get the device factory name

- int get_device_retransmission (int device_id)

  get_device_retransmission Get the device retransmission count on bus

- int get_device_timeout_ms (int device_id)

  get_device_timeout_ms Get The device timeout (ms), The bus have send a package have wait timeout count.

- int get_device_address_len (int device_id)

  get_device_address_len Get the device address length.

- device_management_t ∗ get_device_management (int device_id)

  get_device_management get the device management pointer

- int device_managemnt_init (void)

  device_managemnt_init The device management modele have a initialize

- int set_read_device_information (const read_device_return_t ∗info, int device_id)

  set_read_device_information bus have get a device information have wirte it.

- int get_read_device_information (read_device_return_t ∗out, int device_id)

  get_read_device_information It's read a device information called by adapter layer.

5.17.1 Detailed Description

Functions to rs485 device create , delete , management.

5.17.2 Typedef Documentation

5.17.2.1 typedef struct **device_management device_management_t**

device define the device management struct

5.17.3 Function Documentation

5.17.3.1 bool check_device_id (int *device_id*) `[inline]`

check_object_id check the object is legal

Parameters

`in`	*device_id*	: The need to check device id.

Returns

if object id is legal return true, and return false.

check_object_id check the object is legal

Parameters

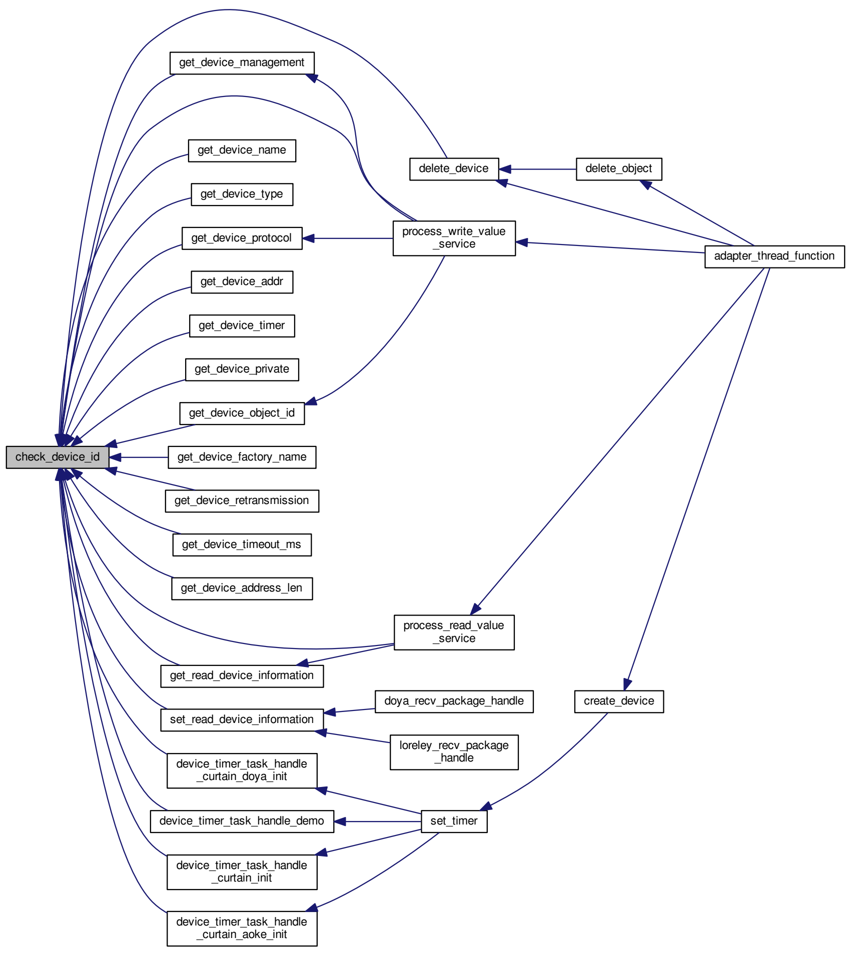
`in`	*device_id*	: The need to check device id.

Returns

if device id is legal return true, and return false.

Definition at line 117 of file device.c.

Here is the caller graph for this function:



5.17.3.2 int create_device (adapter_t * *adapter*)

create_device create a rs485 device, mount the device to protocol

Parameters

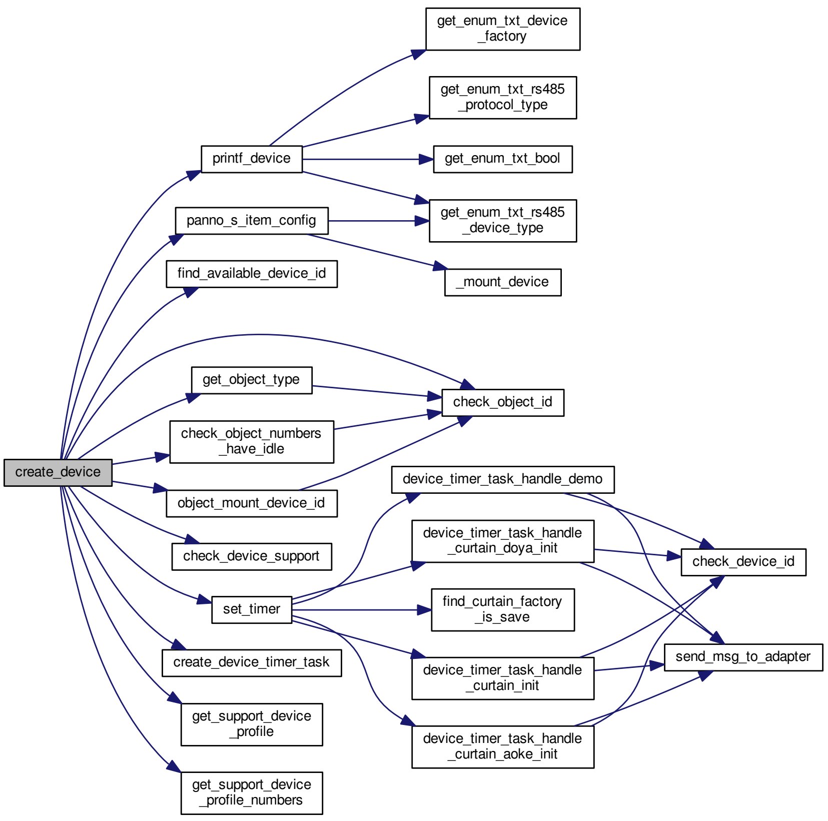
in	*adapter*	: The adapter message service type

Returns

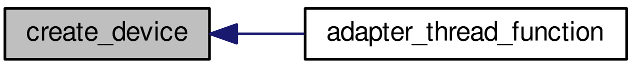
return the device id. if the device id have a negative value, you have create device fail.

Definition at line 214 of file device.c.

Here is the call graph for this function:


get_enum_txt_device_factory
get_enum_txt_rs485_protocol_type
printf_device
get_enum_txt_bool
panno_s_item_config
get_enum_txt_rs485_device_type
find_available_device_id
_mount_device
get_object_type
check_object_id
check_object_numbers_have_idle
create_device
object_mount_device_id
device_timer_task_handle_demo
device_timer_task_handle_curtain_doya_init
check_device_support
check_device_id
set_timer
find_curtain_factory_is_save
create_device_timer_task
device_timer_task_handle_curtain_init
send_msg_to_adapter
get_support_device_profile
device_timer_task_handle_curtain_aoke_init
get_support_device_profile_numbers

Here is the caller graph for this function:



5.17.3.3 int delete_device (int *object_id,* int *device_id*)

delete_device delete a device form device management table.

Parameters

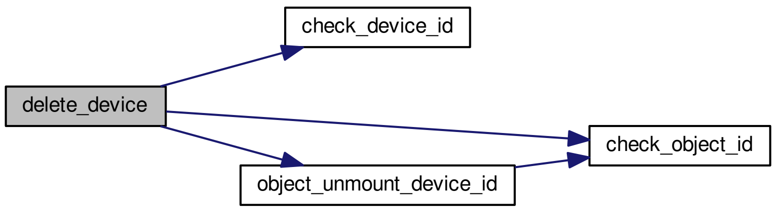
`in`	*device_id*	: The device id .
`in`	*object_id*	: The object id .

Returns

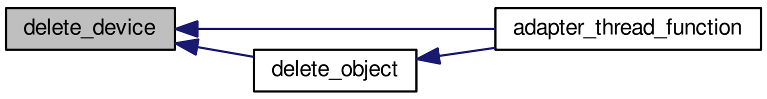
0 is success, others is fail.

Definition at line 380 of file device.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.3.4 int device_managemnt_init (void)

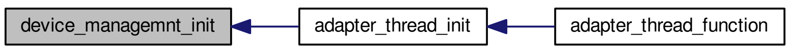
device_managemnt_init The device management modele have a initialize

Returns

0 is success, others is fail.

Definition at line 559 of file device.c.

Here is the caller graph for this function:



5.17.3.5 static int find_available_device_id (void) [inline], [static]

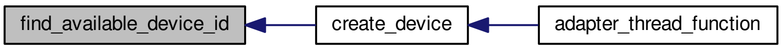
find_available_device_id find a available device ID

Returns

return the device id, but The not positive value is a error.

Definition at line 88 of file device.c.

Here is the caller graph for this function:



5.17.3.6 int get_device_addr (unsigned char * *addr*, unsigned int *addr_len*, int *device_id*) [inline]

get_device_addr get a rs485 device addr, you maybe have no address for some device.

Parameters

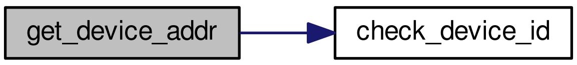
in,out	*addr*	: The device address pointer
in	*addr_len*	: The device address buffer length.
in	*device_id*	: The device id.

Returns

0 is success, and others is fail.

Definition at line 447 of file device.c.

Here is the call graph for this function:



5.17.3.7 int get_device_address_len (int *device_id*) [inline]

get_device_address_len Get the device address length.

Parameters

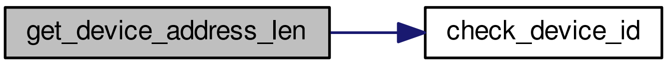
`in`	*device_id*	: The device id.

Returns

The device address len, if have a error return negative value.

Definition at line 536 of file device.c.

Here is the call graph for this function:



5.17.3.8 int get_device_factory_name (int *device_id*) `[inline]`

get_device_factory_name Get the device factory name

Parameters

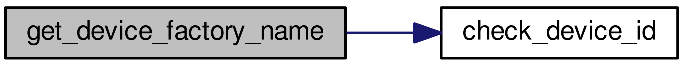
`in`	*device_id*	: The device id.

Returns

The device factory name numbers define by enum.h

Definition at line 503 of file device.c.

Here is the call graph for this function:



5.17.3.9 device_management_t∗ get_device_management (int *device_id*)

get_device_management get the device management pointer

Parameters

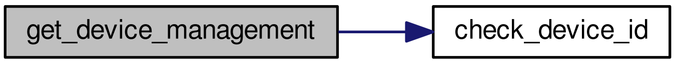
device_id	: The device id

Returns

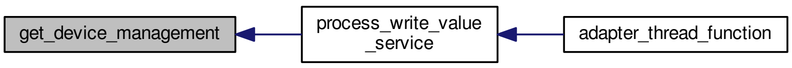
The device management pointer, if error, and return NULL.

Definition at line 547 of file device.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.3.10 int get_device_name (char * *out,* int *out_len,* int *device_id*) [inline]

get_device_name get a device name from device database.

Parameters

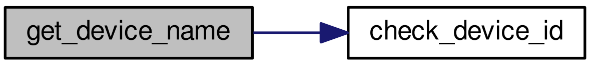
in,out	*out*	: The device name have write it.
in	*out_len*	: The devide name buffer length.
in	*device_id*	: The device id.

Returns

0 is success, and others is fail.

Definition at line 410 of file device.c.

Here is the call graph for this function:



5.17.3.11 int get_device_object_id (int *device_id*) `[inline]`

get_device_object_id get the object id by device id

Parameters

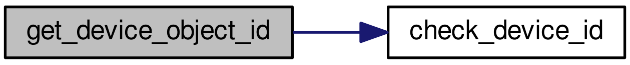
`in`	*device_id*	: The device id.

Returns

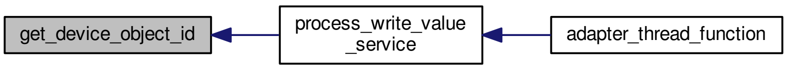
The object id, if have a error return negative value.

Definition at line 491 of file device.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.3.12 struct **device_profile**∗ get_device_private (int *device_id*)

get_device_private get a device private profile

Parameters

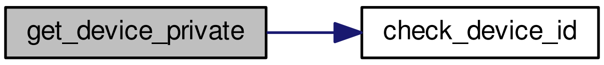
`in`	*device_id*	: The device id.

Returns

The private ppointer, if error ,and return NULL. FIXME: the private pointer is have memcopy a buffer, so, the struct have used?

Definition at line 475 of file device.c.

Here is the call graph for this function:



5.17.3.13 int get_device_private_numbers (int *device_id*) `[inline]`

get_device_private_numbers

Parameters

in	*device_id*	: The device id

Returns

The private profile numbers, if error, and return negative value.

Definition at line 486 of file device.c.

5.17.3.14 int get_device_protocol (int *device_id*) `[inline]`

get_device_protocol get a device protocol from device database, just like BACnet, MODUBS...

Parameters

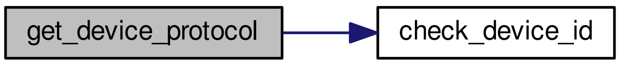
in	*device_id*	: The device id.

Returns

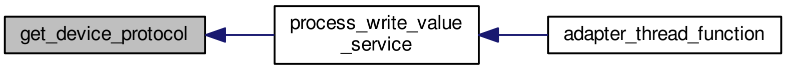
0 is success, and others is fail.

Definition at line 434 of file device.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.3.15 int get_device_retransmission (int *device_id*) [inline]

get_device_retransmission Get the device retransmission count on bus

Parameters

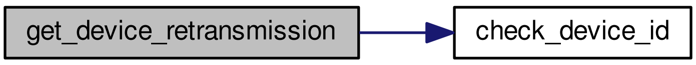
in	*device_id*	: The device id.

Returns

The device retransmission numbers, if have a error return negative value.

Definition at line 514 of file device.c.

Here is the call graph for this function:



5.17.3.16 int get_device_timeout_ms (int *device_id*) [inline]

get_device_timeout_ms Get The device timeout (ms), The bus have send a package have wait timeout count.

Parameters

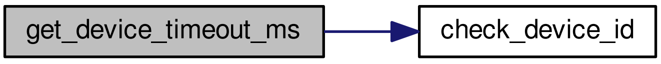
in	*device_id*	: The device id.

Returns

The device timeout, if have a error return negative value.

Definition at line 525 of file device.c.

Here is the call graph for this function:



5.17.3.17 timer_task_t∗ get_device_timer (int *device_id*) `[inline]`

get_device_timer get a device timer task.

Parameters

`in`	*device_id*	: The devcie id.

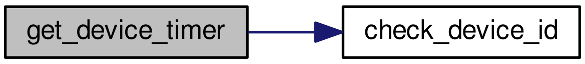
Returns

0 is success, and others is fail.

FIXME: the timer pointer is have memcopy a buffer, so, the struct have used?

Definition at line 463 of file device.c.

Here is the call graph for this function:



5.17.3.18 int get_device_type (int *device_id*) `[inline]`

get_device_type get a device type from device database, just like air condition, fresh air.....

Parameters

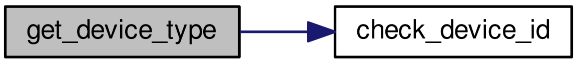
`in`	*device_id*	: The device id.

Returns

0 is success, and others is fail.

Definition at line 423 of file device.c.

Here is the call graph for this function:



5.17.3.19 int get_read_device_information (read_device_return_t ∗ *out*, int *device_id*)

get_read_device_information It's read a device information called by adapter layer.

Parameters

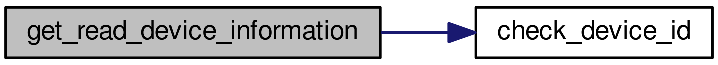
`out`	*out*	: The device private information
`in`	*device_id*	: The device id.

Returns

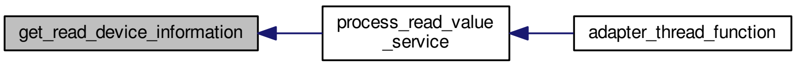
0 is success, others is fail.

Definition at line 586 of file device.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.17.3.20 int set_read_device_information (const read_device_return_t $*$ *info,* int *device_id*)

set_read_device_information bus have get a device information have wirte it.

Parameters

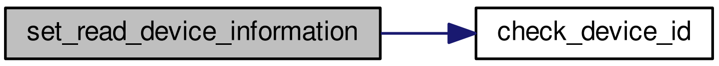
`in`	*info*	: The device private information
`in`	*device_id*	: The device id.

Returns

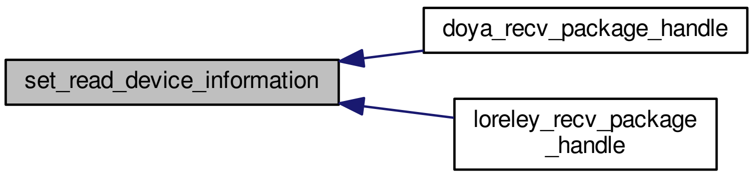
0 is success, others is fail.

Definition at line 571 of file device.c.

Here is the call graph for this function:

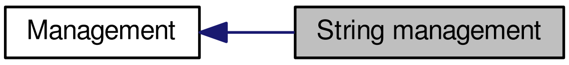


Here is the caller graph for this function:



5.18 String management

Collaboration diagram for String management:



Functions

- char ∗ get_enum_txt_service (rs485_service_type_enum type)

  get_enum_txt_service get enum rs485 service message type
- char ∗ get_enum_txt_rs485_device_type (rs485_device_type_enum type)

  get_enum_txt_rs485_device_type get enum rs485 device type
- char ∗ get_enum_txt_rs485_protocol_type (rs485_protocol_type_enum type)

  get_enum_txt_rs485_protocol_type get enum rs485 protocol type
- char ∗ get_enum_txt_device_method (rs485_device_method_enum type)

  get_enum_txt_device_method get enum device method(command)
- char ∗ get_enum_txt_device_factory (rs485_factory_name_enum name)

  get_enum_txt_device_factory get enum device factory name
- char ∗ get_enum_txt_bool (bool status)

  get_enum_txt_bool get the string about bool value

5.18.1 Detailed Description

Functions to rs485 service enum to string.

5.18.2 Function Documentation

5.18.2.1 char∗ get_enum_txt_bool (bool *status*)

get_enum_txt_bool get the string about bool value

Parameters

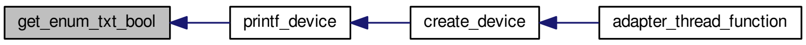
in	*status*	: The bool status

Returns

a string about the true and false value.

Definition at line 235 of file enumtxt.c.

Here is the caller graph for this function:



5.18.2.2 char∗ get_enum_txt_device_factory (rs485_factory_name_enum *name*)

get_enum_txt_device_factory get enum device factory name

Parameters

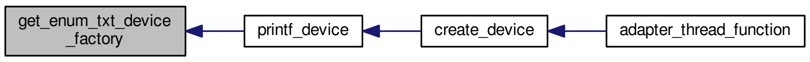
`in`	*name*	: The rs485 device factory name , define on enum.h

Returns

a string about the device factory name

Definition at line 210 of file enumtxt.c.

Here is the caller graph for this function:



5.18.2.3 char∗ get_enum_txt_device_method (rs485_device_method_enum *type*)

get_enum_txt_device_method get enum device method(command)

Parameters

`in`	*type*	: rs485 device method type , define on enum.h

Returns

a string about the device command

Definition at line 95 of file enumtxt.c.

5.18.2.4 char∗ get_enum_txt_rs485_device_type (rs485_device_type_enum *type*)

get_enum_txt_rs485_device_type get enum rs485 device type

Parameters

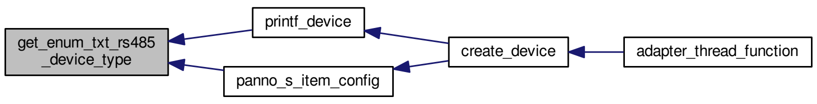
in	*type*	: rs485 device type, define on enum.h

Returns

a string about the device type

Definition at line 56 of file enumtxt.c.

Here is the caller graph for this function:



5.18.2.5 char∗ get_enum_txt_rs485_protocol_type (rs485_protocol_type_enum *type*)

get_enum_txt_rs485_protocol_type get enum rs485 protocol type

Parameters

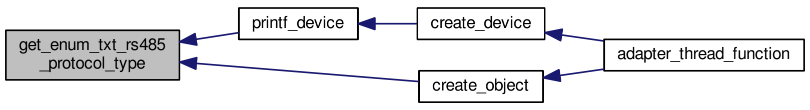
in	*type*	: rs485 protocol type, define on enum.h

Returns

a string about the rs485 protocol type

Definition at line 76 of file enumtxt.c.

Here is the caller graph for this function:



5.18.2.6 char∗ get_enum_txt_service (rs485_service_type_enum *type*)

get_enum_txt_service get enum rs485 service message type

Parameters

in	*type*	: rs485 service type , define on enum.h

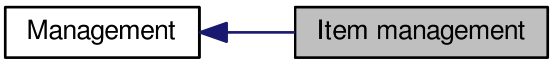
Returns

a string about the message

Definition at line 29 of file enumtxt.c.

5.19 Item management

Collaboration diagram for Item management:



Macros

- #define PANNO_S_ITEM_CONFIG
- #define PANNO_S_ITEM_DEFAULT (1)
- #define PANNO_S_ITEM_WENRUDE (0)
- #define PANNO_S_ITEM_ARMANI (0)
- #define PANNO_S_ITEM_SHAOCHENGGUOJI (0)

Functions

- void panno_s_item_config (adapter_t ∗adapter, rs485_device_type_enum device_type, unsigned char device_addr)

  panno_s_item_config This function is offter the pannoS item config

5.19.1 Detailed Description

This moduble have offter the item configure.

```
Default config is : user defiend protocol,

The different item have a different device, so you need to configure it.
```

5.19.2 Macro Definition Documentation

5.19.2.1 #define PANNO_S_ITEM_ARMANI (0)

The Chengdu armani item configure

Definition at line 72 of file item_config.h.

5.19.2.2 #define PANNO_S_ITEM_CONFIG

The pannoS item define , if have no define it, This information have write by client.

Definition at line 42 of file item_config.h.

5.19.2.3 #define PANNO_S_ITEM_DEFAULT (1)

The pannoS default item confiure

Definition at line 66 of file item_config.h.

5.19.2.4 #define PANNO_S_ITEM_SHAOCHENGGUOJI (0)

The Chengdu shaochengguoji item configure

Definition at line 75 of file item_config.h.

5.19.2.5 #define PANNO_S_ITEM_WENRUDE (0)

The Chengdu wenrude item configure

Definition at line 69 of file item_config.h.

5.19.3 Function Documentation

5.19.3.1 void panno_s_item_config (**adapter_t** ∗ *adapter,* **rs485_device_type_enum** *device_type,* unsigned char *device_addr*)

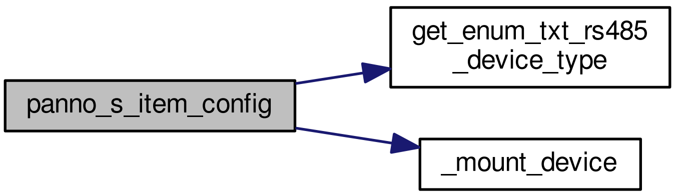
panno_s_item_config This function is offter the pannoS item config

Parameters

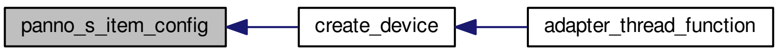
`in,out`	*adapter*	: The adapter struct, have write some device information in it.
`in`	*device_type*	: The device type, air condition/curtain/fresh air/...
`in`	*device_addr*	: The device addr, just for pannoS KNX 1 byte address.

Definition at line 157 of file item_config.c.

Here is the call graph for this function:

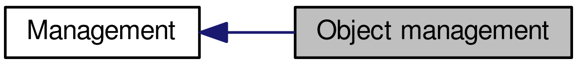


Here is the caller graph for this function:



5.20 Object management

Collaboration diagram for Object management:



Data Structures

- struct object_management

    object_management define the object management struct

Typedefs

- typedef struct object_management object_management_t

    object_management define the object management struct

Functions

- bool check_object_id (int object_id)

    check_object_id check the object is legal
- static int find_available_object_id (void)

    find_available_object_id Find a available object id from object table
- static bool object_is_used (const adapter_t ∗adapter)

    object_is_used To determine whether the object id has been used
- static int work_thread_create (object_management_t ∗object)

    work_thread_create create work thread
- static void work_thread_clean (object_management_t ∗object)

    work_thread_clean clean the work thread
- int create_object (const adapter_t ∗adapter)

    create_object create a object by the adapter message
- int delete_object (int object_id)

    delete_object delete a rs485 object by object id
- int get_object_type (int object_id)

    get_object_type get the object protocol type
- int get_object_mount_device (int object_id, int ∗out_id, int out_id_len)

    get_object_mount_device get the object mount device
- bool check_object_numbers_have_idle (int object_id)

    check_object_numbers_have_idle check object mount device is full ?
- int object_mount_device_id (int object_id, int device_id)

    object_mount_device_id add a device to his object
- void object_unmount_device_id (int object_id, int device_id)

    object_unmount_device_id delete a device form his object
- void ∗ get_object_work_queue (int object_id)

get_object_work_queue get the object of work queue

- void ∗ get_object_queue_sem (int object_id)

  get_object_queue_sem get the object of work queue semphore

5.20.1 Detailed Description

Functions to rs485 Object create ,delete, management.

```
The object It's consist of the rs485 protocol .
The Modbus is a object,
The BACnet is a object,


every object have create a work thread to process the work.
```

5.20.2 Typedef Documentation

5.20.2.1 typedef struct **object_management object_management_t**

object_management define the object management struct

5.20.3 Function Documentation

5.20.3.1 bool check_object_id (int *object_id*) [inline]

check_object_id check the object is legal

Parameters

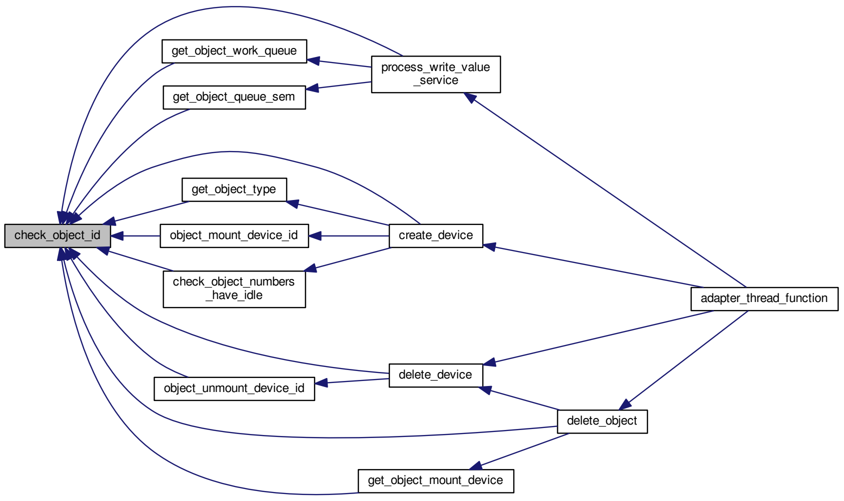
in	*object_id*	: The need to check object id.

Returns

if object id is legal return true, and return false.

Definition at line 74 of file object.c.

Here is the caller graph for this function:



5.20.3.2 bool check_object_numbers_have_idle (int *object_id*)

check_object_numbers_have_idle check object mount device is full ?

Parameters

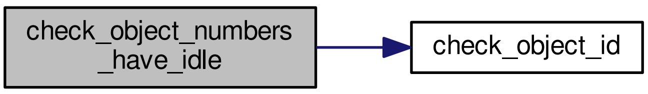
`in`	*object_id*	: The object id

Returns

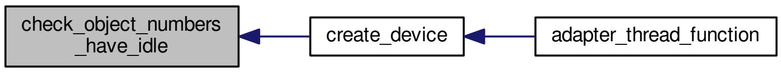
If the object have mount device have not full return true, or return false.

Definition at line 535 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.3 int create_object (const adapter_t * *adapter*)

create_object create a object by the adapter message

Parameters

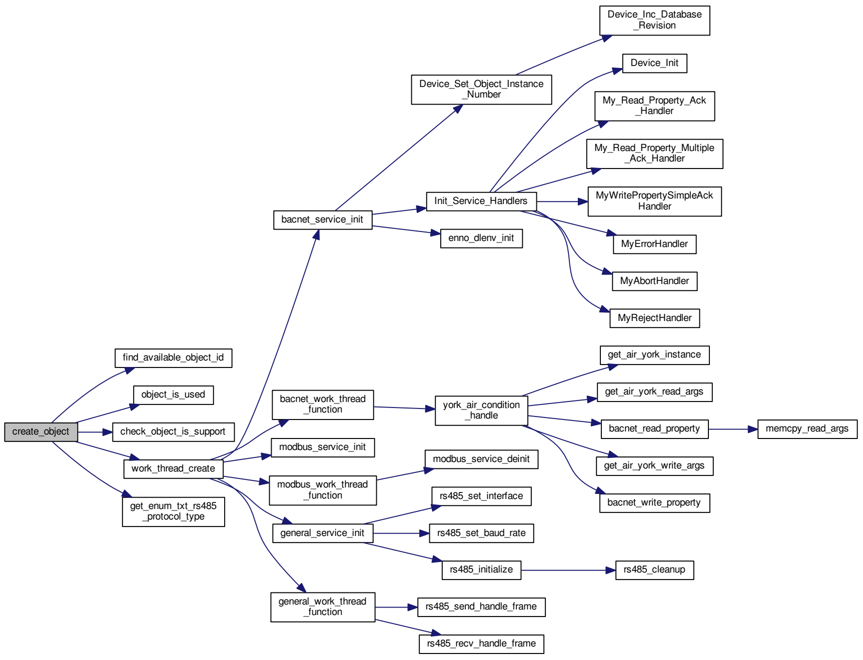
in	*adapter*	: The adapter message

Returns

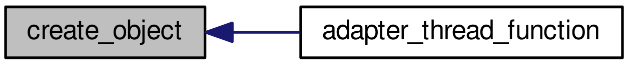
0 is success, others is fail.

Definition at line 326 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.4 int delete_object (int *object_id*)

delete_object delete a rs485 object by object id

Parameters

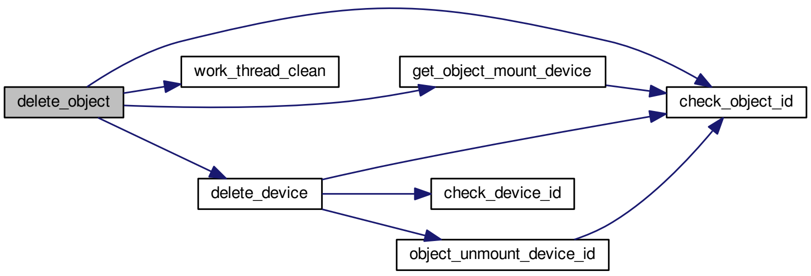
in	*object_id*	: The you want to delete object id.

Returns

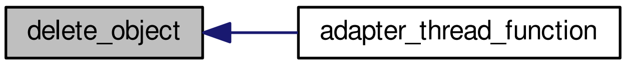
0 is success, others is fail.

Definition at line 421 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.5 static int find_available_object_id (void) [inline], [static]

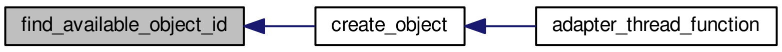
find_available_object_id Find a available object id from object table

Returns

return the available object id, or return negative

Definition at line 96 of file object.c.

Here is the caller graph for this function:



5.20.3.6 int get_object_mount_device (int *object_id,* int ∗ *out_id,* int *out_id_len*)

get_object_mount_device get the object mount device

Parameters

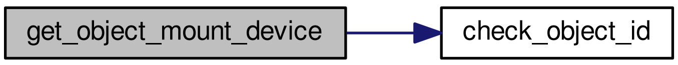
in	*object_id*	: The object id
out	*out_id*	: Out the device id on this object
in	*out_id_len*	: The out buffer length.

Returns

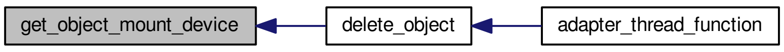
return the mount device numbers ,negative value is error

Definition at line 467 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.7 void∗ get_object_queue_sem (int *object_id*)

get_object_queue_sem get the object of work queue semphore

Parameters

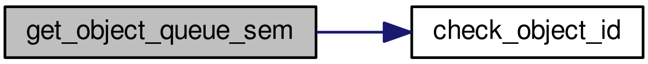
`in`	*object_id*	: The object id

Returns

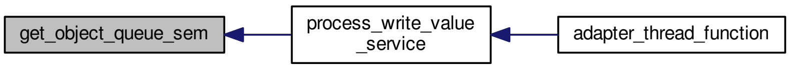
The object of work queue semphore pointer, or return NULL.

Definition at line 564 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.8 int get_object_type (int *object_id*)

get_object_type get the object protocol type

Parameters

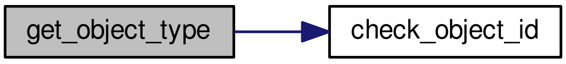
`in`	*object_id*	: The object id

Returns

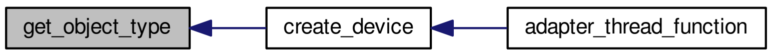
return the protocol type, return a negative value is error

Definition at line 455 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.9 void∗ get_object_work_queue (int *object_id*)

get_object_work_queue get the object of work queue

Parameters

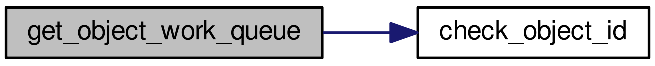
in	*object_id*	: The object id

Returns

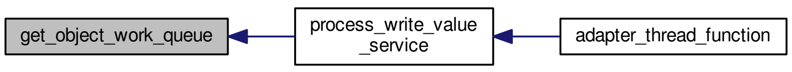
The object of work queue pointer, or return NULL.

Definition at line 554 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.10 static bool object_is_used (const adapter_t ∗ *adapter*) [inline], [static]

object_is_used To determine whether the object id has been used

Parameters

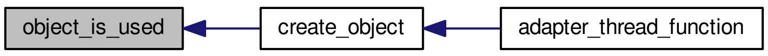
adapter	: The adapter struct.

Returns

if have used return false, and return ture;

Definition at line 127 of file object.c.

Here is the caller graph for this function:



5.20.3.11 int object_mount_device_id (int *object_id,* int *device_id*)

object_mount_device_id add a device to his object

Parameters

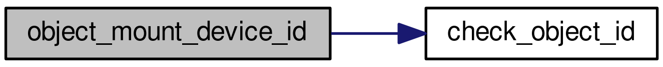
in	*object_id*	: The object id
in	*device_id*	: The device id

Returns

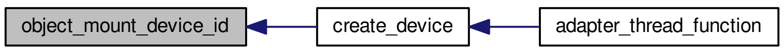
0 is success, others is fail.

Definition at line 493 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.12 void object_unmount_device_id (int *object_id,* int *device_id*)

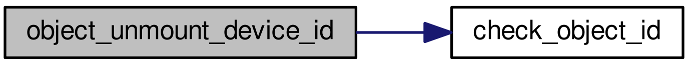
object_unmount_device_id delete a device form his object

Parameters

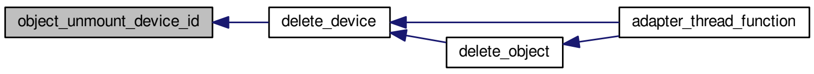
in	*object_id*	: The object id
in	*device_id*	: The device id

Definition at line 516 of file object.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.20.3.13 static void work_thread_clean (object_management_t ∗ *object*) [static]

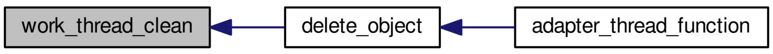
work_thread_clean clean the work thread

Parameters

in	*object*	: The object device_management_t struct.

Definition at line 305 of file object.c.

Here is the caller graph for this function:



5.20.3.14 static int work_thread_create (object_management_t ∗ *object*) [static]

work_thread_create create work thread

Parameters

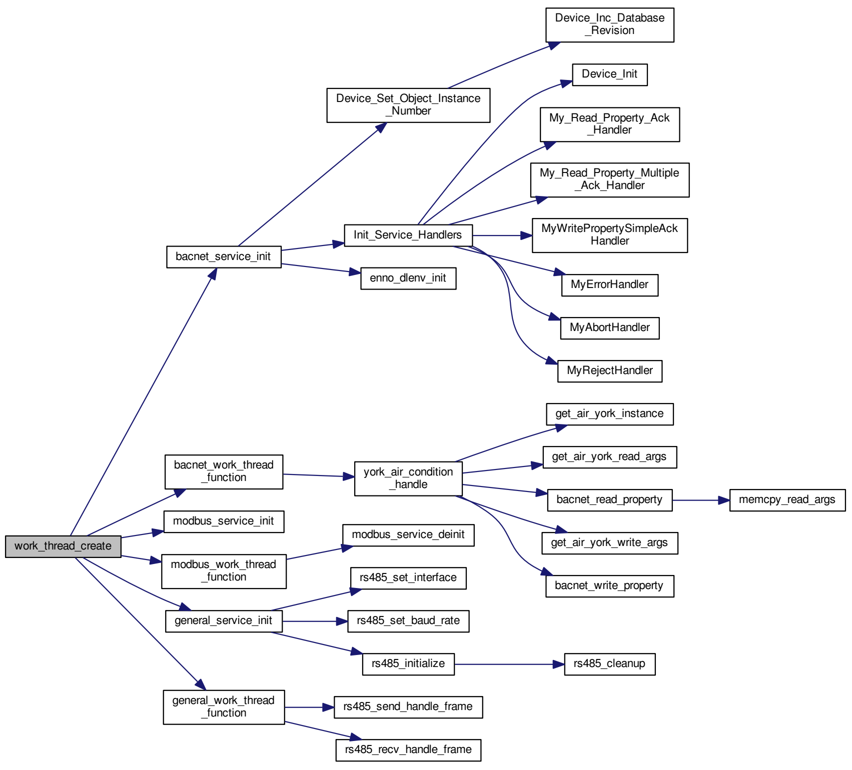
`in,out`	*object*	: The object struct information

Returns

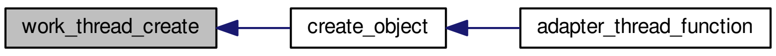
0 is create success, others is fail.

Definition at line 158 of file object.c.

Here is the call graph for this function:

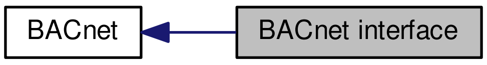


Here is the caller graph for this function:



5.21 BACnet interface

Collaboration diagram for BACnet interface:



Data Structures

- struct bacnet

    bacnet bacnet interface struct

Typedefs

- typedef struct bacnet bacnet_port_handle_t

    bacnet bacnet interface struct

Functions

- void ∗ bacnet_work_thread_function (void ∗arg)

    bacnet_work_thread_function The bacnet work thread

5.21.1 Detailed Description

define the bacnet interface

5.21.2 Typedef Documentation

5.21.2.1 typedef struct bacnet bacnet_port_handle_t

bacnet bacnet interface struct

5.21.3 Function Documentation

5.21.3.1 void∗ bacnet_work_thread_function (void ∗ *arg*)

bacnet_work_thread_function The bacnet work thread

Parameters

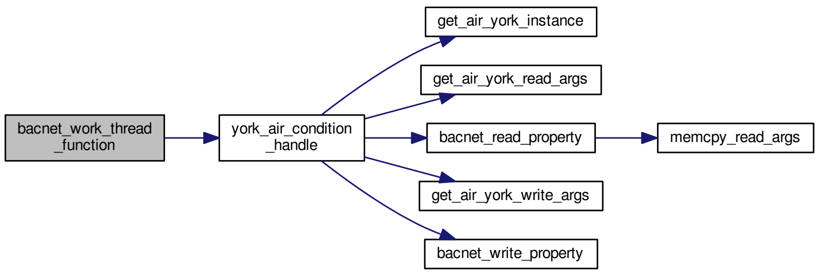
`in`	*arg*	: The object management pointer

Returns

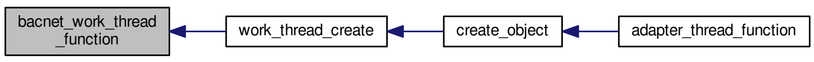
have no return.

Definition at line 107 of file bacnet.c.

Here is the call graph for this function:

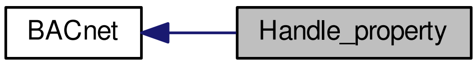


Here is the caller graph for this function:



5.22 Handle_property

Collaboration diagram for Handle_property:



Data Structures

- struct bacnet_write_args_t

  bacnet write arg struct
- struct bacnet_read_args_t

  bacnet read property struct

Macros

- #define BACNET_READ_ARGS_OBJECT_MAX 10

Functions

- int get_air_condition_bacnet_write_args (bacnet_write_args_t ∗args, unsigned int device_id, int command)

  get_air_condition_bacnet_write_args bacnet write args
- int get_air_condition_bacnet_read_args (bacnet_read_args_t ∗args, unsigned int device_id)

  get_air_condition_bacnet_read_args bacnet read args
- int bacnet_service_init (object_management_t ∗adapter)

  bacnet_service_init bacnet physics initialze.

5.22.1 Detailed Description

Function of BACnet handle property.

5.22.2 Macro Definition Documentation

5.22.2.1 #define BACNET_READ_ARGS_OBJECT_MAX 10

Definition at line 51 of file handle_property.h.

5.22.3 Function Documentation

5.22.3.1 int bacnet_service_init (**object_management_t** ∗ *adapter*)

bacnet_service_init bacnet physics initialze.

Parameters

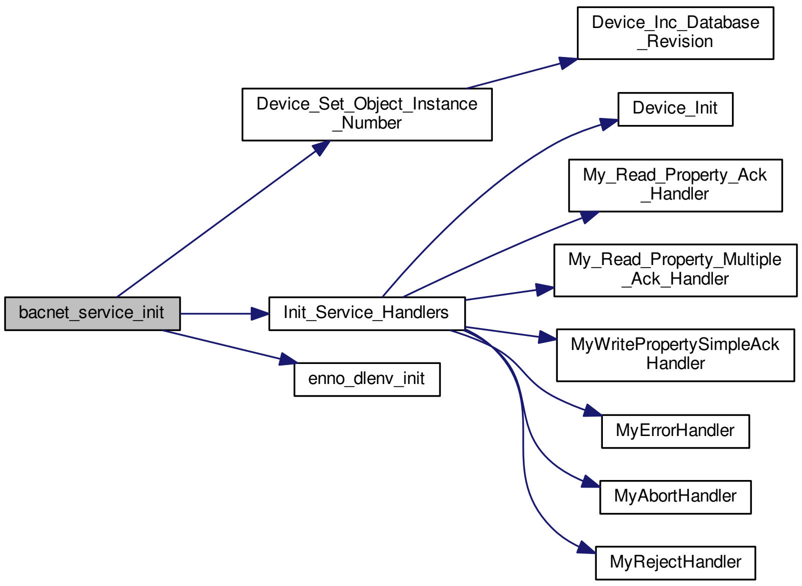
`in`	*adapter*	: bacnet work thread initial.

Returns

0 is success, others is fail.

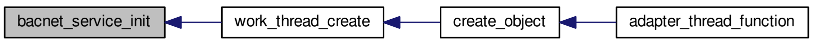
Definition at line 149 of file handle_property.c.

Here is the call graph for this function:



Device_Inc_Database _Revision
Device_Set_Object_Instance _Number
Device_Init
My_Read_Property_Ack _Handler
My_Read_Property_Multiple _Ack_Handler
bacnet_service_init
Init_Service_Handlers
MyWritePropertySimpleAck Handler
enno_dlenv_init
MyErrorHandler
MyAbortHandler
MyRejectHandler

Here is the caller graph for this function:



5.22.3.2 int get_air_condition_bacnet_read_args (bacnet_read_args_t ∗ *args*, unsigned int *device_id*)

get_air_condition_bacnet_read_args bacnet read args

Parameters

`in`	*args*	: bacnet read args struct
`in`	*device_id*	: bacnet device id

Returns

0 is success, others is fail.

5.22.3.3 int get_air_condition_bacnet_write_args (**bacnet_write_args_t** $*$ *args,* unsigned int *device_id,* int *command*)

get_air_condition_bacnet_write_args bacnet write args

Parameters

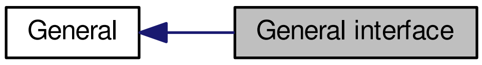
`in`	*args*	: bacnet write args struct
`in`	*device_id*	: bacnet device id
`in`	*command*	: bacnet air command

Returns

0 is success, others is fail.

5.23 General interface

Collaboration diagram for General interface:



Data Structures

- struct mstp_port_handle

  mstp_port_handle general protocol(user defined)

Typedefs

- typedef struct mstp_port_handle mstp_port_handle_t

  mstp_port_handle general protocol(user defined)

Functions

- int general_service_init (object_management_t ∗object)

  general_service_init The general protocol(user defined) initilize
- void ∗ general_work_thread_function (void ∗arg)

  general_work_thread_function The general work thread function

5.23.1 Detailed Description

define the general interface interface

5.23.2 Typedef Documentation

5.23.2.1 typedef struct mstp_port_handle mstp_port_handle_t

mstp_port_handle general protocol(user defined)

5.23.3 Function Documentation

5.23.3.1 int general_service_init (object_management_t ∗ *object*)

general_service_init The general protocol(user defined) initilize

Parameters

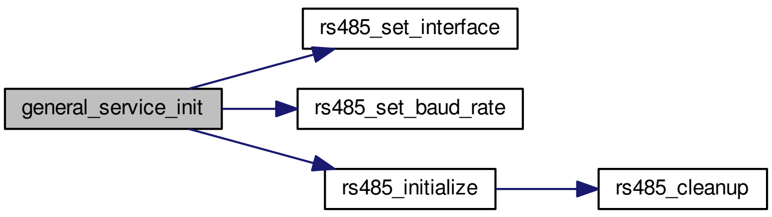
`in`	*object*	: The object information.

Returns

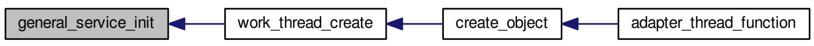
0 is success, others is fail.

Definition at line 46 of file general.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.23.3.2 void∗ general_work_thread_function (void ∗ *arg*)

general_work_thread_function The general work thread function

Parameters

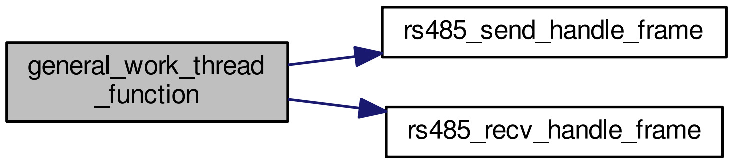
`in`	*arg*	: The thread argument , This arg uesd "The general object, mstp_port_←handle_t"

Returns

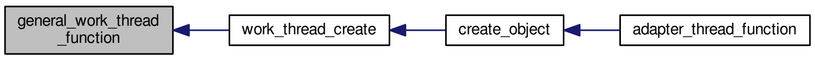
Thread have error have a return.

Definition at line 67 of file general.c.

Here is the call graph for this function:

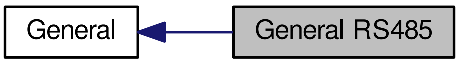


Here is the caller graph for this function:



5.24 General RS485

Collaboration diagram for General RS485:



Functions

- void rs485_set_interface (char ∗ifname)

  RS485_Set_Interface rs485 interface name.
- const char ∗ rs485_get_interface (void)

  RS485_Get_Interface get the rs485 interface name.
- void rs485_initialize (void)

  RS485_Initialize.
- int rs485_send_handle_frame (volatile struct mstp_port_handle ∗mstp_port)

  rs485_send_handle_frame rs485 bus package a send frame, and send the package to bus.
- int rs485_recv_handle_frame (volatile struct mstp_port_handle ∗mstp_port)

  rs485_recv_handle_frame rs485 bus receive a frame, and call process these data.
- bool rs485_set_baud_rate (uint32_t baud)

  RS485_Set_Baud_Rate set the rs485 buad rate.
- void rs485_cleanup (void)

  RS485_Cleanup The rs485 initaialize fail, have clean.

5.24.1 Detailed Description

define the rs485 physics driver interface

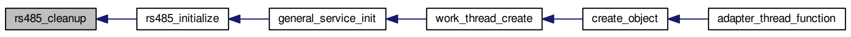
5.24.2 Function Documentation

5.24.2.1 void rs485_cleanup (void)

RS485_Cleanup The rs485 initaialize fail, have clean.

Definition at line 401 of file rs485.c.

Here is the caller graph for this function:



5.24.2.2 const char∗ rs485_get_interface (void)

RS485_Get_Interface get the rs485 interface name.

Returns

The rs485 interface name

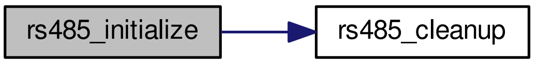
5.24.2.3 void rs485_initialize (void)

RS485_Initialize.

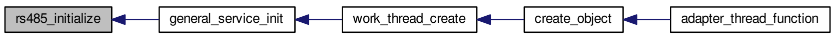
initiallize a rs485 interface

Definition at line 411 of file rs485.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.24.2.4 int rs485_recv_handle_frame (volatile struct **mstp_port_handle** ∗ *mstp_port*)

rs485_recv_handle_frame rs485 bus receive a frame, and call process these data.

Parameters

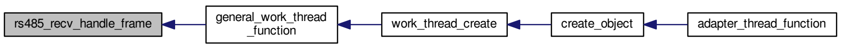
in	*mstp_port*	: The mstp_port struct, and consist of some frame infromation

Returns

0 is success, and others is fail.

Definition at line 331 of file rs485.c.

Here is the caller graph for this function:



5.24.2.5 int rs485_send_handle_frame (volatile struct mstp_port_handle * *mstp_port*)

rs485_send_handle_frame rs485 bus package a send frame, and send the package to bus.

Parameters

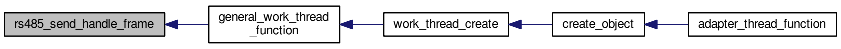
in	*mstp_port*	: The mstp_port struct, and consist of some send frame information

Returns

0 is success, and others is fail.

Definition at line 281 of file rs485.c.

Here is the caller graph for this function:



5.24.2.6 bool rs485_set_baud_rate (uint32_t *baud*)

RS485_Set_Baud_Rate set the rs485 buad rate.

Parameters

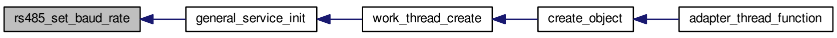
in	*baud*	: The rs485 UART buad, like "B9600" "B38400"

Returns

True is set ok, and others is set fail.

Definition at line 200 of file rs485.c.

Here is the caller graph for this function:



5.24.2.7 void rs485_set_interface (char ∗ *ifname*)

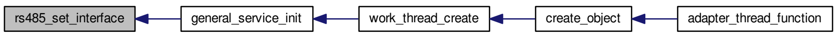
RS485_Set_Interface rs485 interface name.

Parameters

in	*ifname*	: The rs485 name , just like "rs4851"

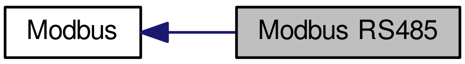
Definition at line 90 of file rs485.c.

Here is the caller graph for this function:



5.25 Modbus RS485

Collaboration diagram for Modbus RS485:



Data Structures

- struct modbus_port_handle_t

  The modbus port interface.

Functions

- void ∗ modbus_work_thread_function (void ∗arg)

  modbus_work_thread_function The modbus work thread
- int modbus_service_init (object_management_t ∗object)

  modbus_service_init The modbus interface intialize.
- void modbus_service_deinit (object_management_t ∗object)

  modbus_service_deinit clean the modbus service, The haved called by thread have exit.

5.25.1 Detailed Description

define the modbus interface

5.25.2 Function Documentation

5.25.2.1 void modbus_service_deinit (object_management_t ∗ *object*)

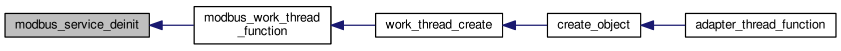
modbus_service_deinit clean the modbus service, The haved called by thread have exit.

Parameters

in	*object*	: The object information

Definition at line 292 of file modbus.c.

Here is the caller graph for this function:



5.25.2.2 int modbus_service_init (**object_management_t** ∗ ***object***)

modbus_service_init The modbus interface intialize.

Parameters

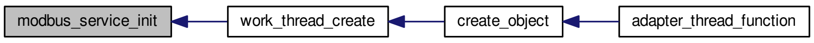
`in`	*object*	: The object port information

Returns

0 is success, others is fail.

Definition at line 254 of file modbus.c.

Here is the caller graph for this function:



5.25.2.3 void∗ modbus_work_thread_function (void ∗ *arg*)

modbus_work_thread_function The modbus work thread

Parameters

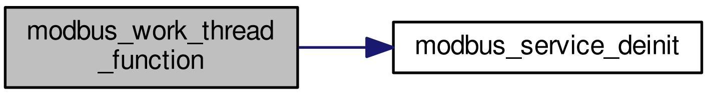
`in`	*arg*	: The thread argument is object management pointer.

Returns

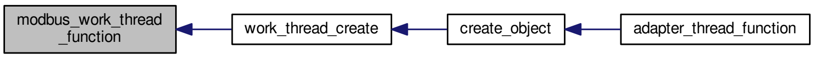
Have no return.

Definition at line 48 of file modbus.c.

Here is the call graph for this function:

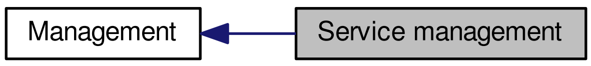


Here is the caller graph for this function:



5.26 Service management

Collaboration diagram for Service management:



Data Structures

- struct thread_pool_t

    define the thread pool struct

Functions

- static int rs485_thread_pool_create (thread_pool_t ∗pool, int numbers)

    rs485_thread_pool_create create linux thread pool
- static void rs485_thread_pool_clean (void)

    rs485_thread_pool_clean clean the linux thread haved create
- void rs485_service_create_clean (void)

    rs485_service_create_clean have clean the rs485 socket communicate
- static int rs485_service_running (const char ∗path)

    rs485_service_running The rs485 service function, It's wait the client requests. It's block
- static void rs485_service_running_clean (void)

    rs485_service_running_clean Have clean the service running
- int rs485_service_start (void)

    rs485_service_start The rs485 service start
- int rs485_send_msg_to_client (int clifd, void ∗buffer, int buffer_len)

    rs485_send_msg_to_client send The message to a client
- int rs485_recv_msg_from_client (int clifd, void ∗buffer, int buffer_len)

    rs485_recv_msg_from_client recvieve a message from client
- int send_msg_to_adapter (const adapter_t ∗adapter)

    send_msg_to_adapter send a message to self,

5.26.1 Detailed Description

Functions to rs485 server have offer the service.

5.26.2 Function Documentation

5.26.2.1 int rs485_recv_msg_from_client (int *clifd,* void ∗ *buffer,* int *buffer_len*)

rs485_recv_msg_from_client recvieve a message from client

Parameters

in	*clifd*	: The client socket descriptor
out	*buffer*	: The buffer have recevie data
in	*buffer_len*	: Teh recevie buffer length

Returns

The receive data length, If have a error have return negative value.

5.26.2.2 int rs485_send_msg_to_client (int *clifd,* void ∗ *buffer,* int *buffer_len*)

rs485_send_msg_to_client send The message to a client

Parameters

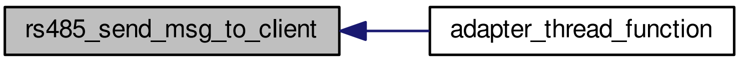
in	*clifd*	: The client socket descriptor
in	*buffer*	: The data buffer you want to send
in	*buffer_len*	: The data buffer length

Returns

The send buffer length, If have a error have return negative value.

Definition at line 246 of file service.c.

Here is the caller graph for this function:



5.26.2.3 void rs485_service_create_clean (void)

rs485_service_create_clean have clean the rs485 socket communicate

Definition at line 233 of file service.c.

5.26.2.4 static int rs485_service_running (const char ∗ *path*) [static]

rs485_service_running The rs485 service function, It's wait the client requests. It's block

Parameters

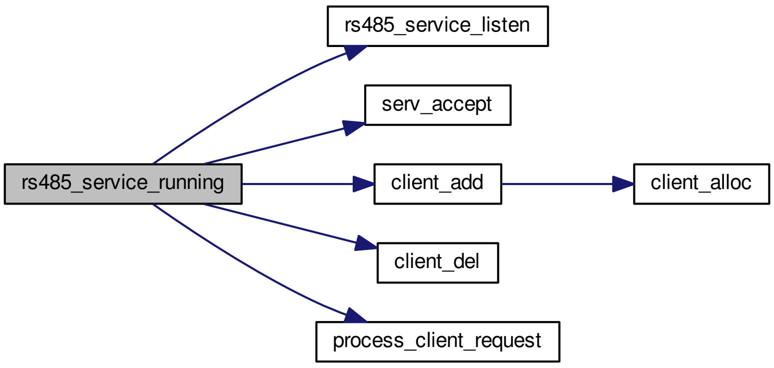
in	*path*	: The communicate unix path.

Returns

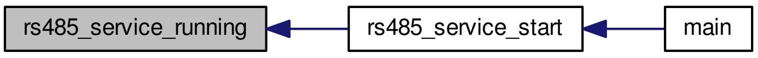
0 is success, and others is fail.

Definition at line 417 of file service.c.

Here is the call graph for this function:



Here is the caller graph for this function:

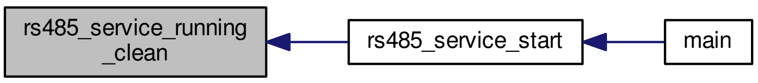


5.26.2.5 static void rs485_service_running_clean (void) [static]

rs485_service_running_clean Have clean the service running

Definition at line 511 of file service.c.

Here is the caller graph for this function:



5.26.2.6 int rs485_service_start (void)

rs485_service_start The rs485 service start

Note

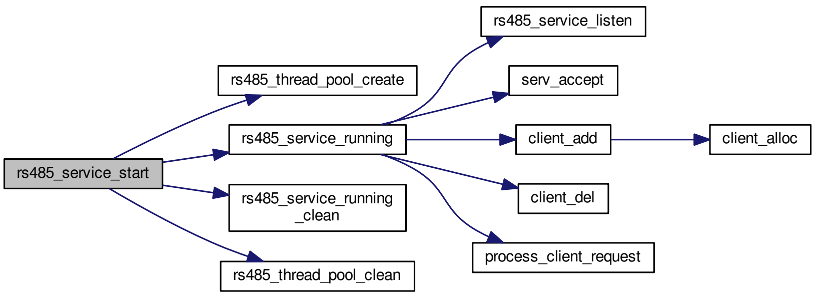
: It's should have not return, until have a error

Returns

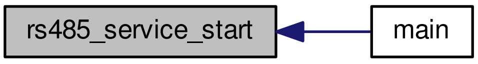
0 is success, and others is fail.

Definition at line 517 of file service.c.

Here is the call graph for this function:



Here is the caller graph for this function:

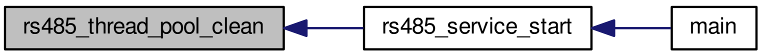


5.26.2.7 static void rs485_thread_pool_clean (void) [static]

rs485_thread_pool_clean clean the linux thread haved create

Definition at line 146 of file service.c.

Here is the caller graph for this function:



5.26.2.8 static int rs485_thread_pool_create (thread_pool_t ∗ *pool,* int *numbers*) [static]

rs485_thread_pool_create create linux thread pool

Parameters

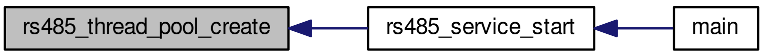
pool[ ]	: The thread pool struct, just statement initialization
numbers	: The thread pool array numbers

Returns

0 is success, others is fial.

Definition at line 110 of file service.c.

Here is the caller graph for this function:



5.26.2.9 int send_msg_to_adapter (const adapter_t ∗ *adapter*)

send_msg_to_adapter send a message to self,

Parameters

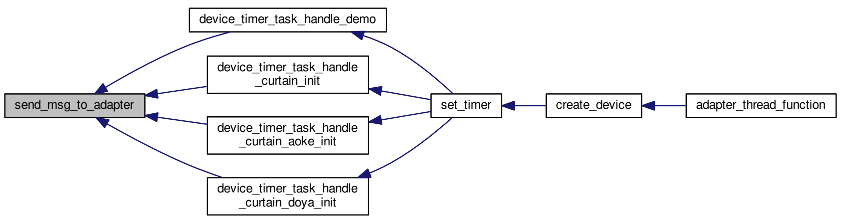
in	*adapter*	: The message struct

Returns

0 is success, and others is fail.

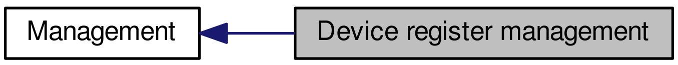
Definition at line 276 of file service.c.

Here is the caller graph for this function:


device_timer_task_handle_demo
device_timer_task_handle _curtain_init
send_msg_to_adapter
set_timer
create_device
adapter_thread_function
device_timer_task_handle _curtain_aoke_init
device_timer_task_handle _curtain_doya_init

5.27 Device register management

Collaboration diagram for Device register management:



Data Structures

- struct device_profile

  device_profile device process method

Typedefs

- typedef int(∗ method_send)(volatile void ∗context)

  int you have full the context pointer.
- typedef int(∗ method_recv)(volatile void ∗context)

  int you have full the context pointer.

Functions

- bool check_device_support (const adapter_t ∗adatper)

  check_device_support check the device have supported by rs485 service
- struct device_profile ∗ get_support_device_profile (rs485_factory_name_enum name)

  get_support_device_profile Get the device profile, The struct device_profile
- int get_support_device_profile_numbers (rs485_factory_name_enum name)

  get_support_device_profile_numbers Get the device profile have support how many command.
- method_send get_device_send_package_function (const struct device_profile ∗profile, int profile_numbers, int command)

  get_device_send_package_function Get the device profile send package callback function
- method_recv get_device_recv_package_function (const struct device_profile ∗profile, int profile_numbers, int command)

  get_device_recv_package_function Get the device profile receive package callback function

5.27.1 Detailed Description

Functions to device register on rs485 service.

5.27.2 Typedef Documentation

5.27.2.1 typedef int(∗ method_recv)(volatile void ∗context)

int you have full the context pointer.

Note

```
if the interface is general, The context = mstp_port_handle_t* handle;
if the interface is bacnet, The context = bacnet_port_handle_t* handle;
if the interface is modbus, The context = modbus_port_handle_t* handle;
```

Parameters

context,The	∗_port_handle_t pointer.

Returns

The send byte numbers, or, return nagetive value

Definition at line 75 of file support.h.

5.27.2.2 typedef int(∗ method_send)(volatile void ∗context)

int you have full the context pointer.

Note

```
if the interface is general, The context = mstp_port_handle_t* handle;
if the interface is bacnet, The context = bacnet_port_handle_t* handle;
if the interface is modbus, The context = modbus_port_handle_t* handle;
```

Parameters

context,The	∗_port_handle_t pointer.

Returns

The send byte numbers, or, return nagetive value

Definition at line 57 of file support.h.

5.27.3 Function Documentation

5.27.3.1 bool check_device_support (const adapter_t ∗ *adatper*)

check_device_support check the device have supported by rs485 service

Parameters

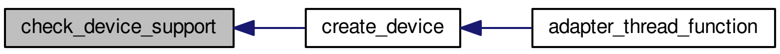
`in`	*adatper*	: The adapter struct, It's define by adapther.h

Returns

If the rs485 service have support this device return true, or return false.

Definition at line 150 of file support.c.

Here is the caller graph for this function:



5.27.3.2 method_recv get_device_recv_package_function (const struct device_profile ∗ *profile,* int *profile_numbers,* int *command*) [inline]

get_device_recv_package_function Get the device profile receive package callback function

Parameters

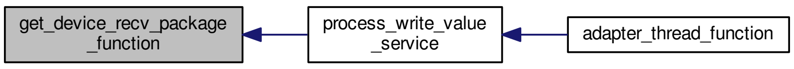
in	*profile*	: The device profile pointer.
in	*profile_numbers*	: The device profile have support command numbers.
in	*command*	: The which command you have chose.

Returns

return the device method recv package callback function pointer, or return NULL.

Definition at line 221 of file support.c.

Here is the caller graph for this function:



5.27.3.3 method_send get_device_send_package_function (const struct device_profile ∗ *profile,* int *profile_numbers,* int *command*) [inline]

get_device_send_package_function Get the device profile send package callback function

Parameters

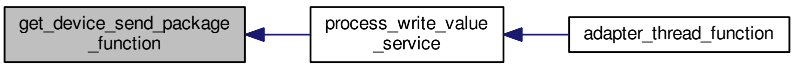
in	*profile*	: The device profile pointer.
in	*profile_numbers*	: The device profile have support command numbers.
in	*command*	: The which command you have chose.

Returns

return the device method send package callback function pointer, or return NULL.

Definition at line 205 of file support.c.

Here is the caller graph for this function:



5.27.3.4 struct device_profile∗ get_support_device_profile (rs485_factory_name_enum *name*)

get_support_device_profile Get the device profile, The struct device_profile

Parameters

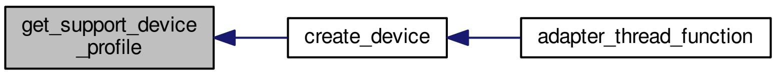
`in`	*name*	: The device factory name enum, It's define by enum.h

Returns

return The device profile pointer, or return NULL.

Definition at line 156 of file support.c.

Here is the caller graph for this function:



5.27.3.5 int get_support_device_profile_numbers (**rs485_factory_name_enum** *name*)

get_support_device_profile_numbers Get the device profile have support how many command.

Parameters

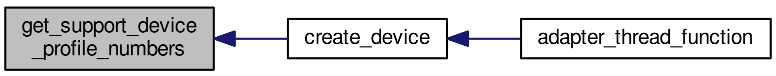
`in`	*name*	: The device factory name enum, It's define by enum.h

Returns

return the numbers about of device support command, or return negative value.

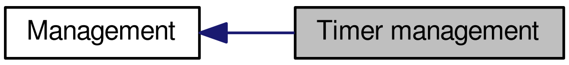
Definition at line 181 of file support.c.

Here is the caller graph for this function:



5.28 Timer management

Collaboration diagram for Timer management:



Data Structures

- struct timer_task_t

  timer task struct

Typedefs

- typedef int(∗ timer_proc_func)(int device_id, int command)

Functions

- void ∗ timer_task_thread_function (void ∗arg)

  timer_task_thread_function The timer task therad start function, just return when the have a error
- int create_device_timer_task (timer_task_t ∗task)

  create_deivce_timer_task create a device timer task , The timer task min tick is 10 second
- int delete_device_timer_task (timer_task_t ∗task)

  delete_device_timer_task delete a device timer task from The timer list.
- int device_timer_task_handle_demo (int device_id, int command)

  device_timer_task_handle_demo timer task handle fucntion demo

5.28.1 Detailed Description

Functions to create or delete the timer task on timer task thread.

5.28.2 Typedef Documentation

5.28.2.1 typedef int(∗ timer_proc_func)(int device_id, int **command**)

Definition at line 32 of file timer_task.h.

5.28.3 Function Documentation

5.28.3.1 int create_device_timer_task (**timer_task_t** ∗ *task*)

create_deivce_timer_task create a device timer task , The timer task min tick is 10 second

Parameters

in	*task*	: The timer task sturct.

Returns

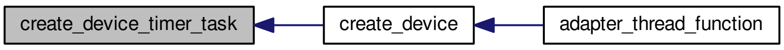
0 is success , others is fail.

Note

The task argument, have used by timer list, so The task struct you must malloc a buffer

Definition at line 153 of file timer_task.c.

Here is the caller graph for this function:



5.28.3.2 int delete_device_timer_task (timer_task_t * *task*)

delete_device_timer_task delete a device timer task from The timer list.

Parameters

in	*task*	: The timer task sturct, you have remote from the list.

Returns

0 is success, others is fail.

Definition at line 196 of file timer_task.c.

5.28.3.3 int device_timer_task_handle_demo (int *device_id,* int *command*)

device_timer_task_handle_demo timer task handle fucntion demo

Parameters

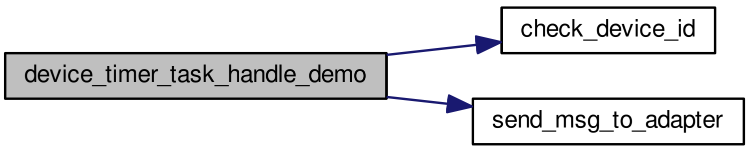
in	*device_id*	: The device Id.
in	*command*	: Get the device information command ,defined by enum.h

Returns

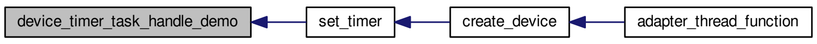
0 is success, others is fail.

Definition at line 222 of file timer_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



5.28.3.4 void∗ timer_task_thread_function (void ∗ *arg*)

timer_task_thread_function The timer task therad start function, just return when the have a error

Parameters

in	*arg*	: The thread argument, unsed.

Returns

: The thread return value.

FIXME : The thread join status, have no set.

Chapter 6

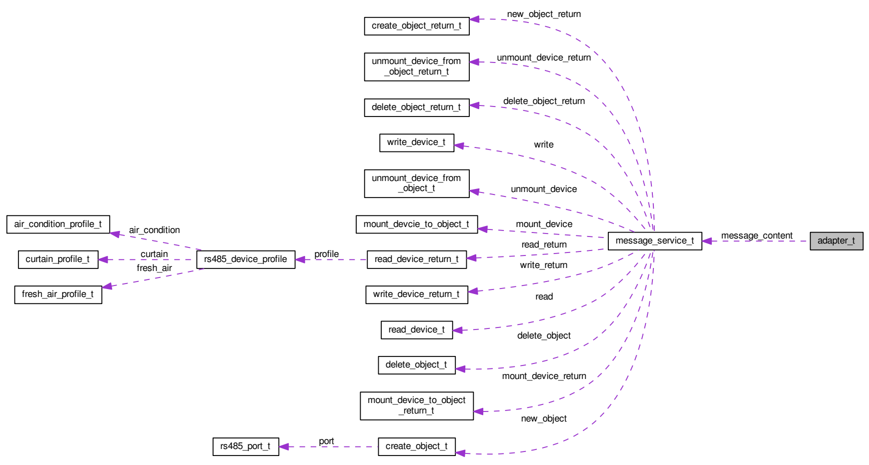
Data Structure Documentation

6.1 adapter_t Struct Reference

define the adapter struct

`#include <adapter.h>`

Collaboration diagram for adapter_t:



Data Fields

- rs485_service_type_enum message_type
- unsigned int message_length
- int message_retvl
- int socket_fd
- message_service_t message_content

6.1.1 Detailed Description

define the adapter struct

Definition at line 313 of file adapter.h.

6.1.2 Field Documentation

6.1.2.1 message_service_t message_content

The message content

Definition at line 324 of file adapter.h.

6.1.2.2 unsigned int message_length

The message length, just like "sizeof(struct adapter_t)", It's used to check the pakcage imperfections

Definition at line 318 of file adapter.h.

6.1.2.3 int message_retvl

The message retvl, just message process retvl, the retvl, have used to client to check the service

Definition at line 320 of file adapter.h.

6.1.2.4 rs485_service_type_enum message_type

The message service type

Definition at line 316 of file adapter.h.

6.1.2.5 int socket_fd

The message socket id

Definition at line 322 of file adapter.h.

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

- include/adapter.h

6.2 air_condition_profile_t Struct Reference

The air conditon profile.

```
#include <adapter.h>
```

Data Fields

- int room_temperature
- int outdoor_temperature
- int pipe_temperature
- int current_mode
- int current_swing
- int current_fan
- int current_set_temperature

6.2.1 Detailed Description

The air conditon profile.

Definition at line 216 of file adapter.h.

6.2.2 Field Documentation

6.2.2.1 int current_fan

Definition at line 223 of file adapter.h.

6.2.2.2 int current_mode

Definition at line 221 of file adapter.h.

6.2.2.3 int current_set_temperature

Definition at line 224 of file adapter.h.

6.2.2.4 int current_swing

Definition at line 222 of file adapter.h.

6.2.2.5 int outdoor_temperature

Definition at line 219 of file adapter.h.

6.2.2.6 int pipe_temperature

Definition at line 220 of file adapter.h.

6.2.2.7 int room_temperature

Definition at line 218 of file adapter.h.

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

- include/adapter.h

6.3 bacnet Struct Reference

bacnet bacnet interface struct

```
#include <bacnet.h>
```

Data Fields

- unsigned char device_mac [4]
- unsigned int mac_length
- rs485_factory_name_enum factory_name

- rs485_device_type_enum device_type
- unsigned int command
- int value
- int value_reserve
- void ∗ arg

6.3.1 Detailed Description

bacnet bacnet interface struct

Definition at line 41 of file bacnet.h.

6.3.2 Field Documentation

6.3.2.1 void∗ arg

Definition at line 50 of file bacnet.h.

6.3.2.2 unsigned int command

Definition at line 47 of file bacnet.h.

6.3.2.3 unsigned char device_mac[4]

Definition at line 43 of file bacnet.h.

6.3.2.4 rs485_device_type_enum device_type

Definition at line 46 of file bacnet.h.

6.3.2.5 rs485_factory_name_enum factory_name

Definition at line 45 of file bacnet.h.

6.3.2.6 unsigned int mac_length

Definition at line 44 of file bacnet.h.

6.3.2.7 int value

Definition at line 48 of file bacnet.h.

6.3.2.8 int value_reserve

Definition at line 49 of file bacnet.h.

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

- include/protocol/bacnet/bacnet.h

6.4 bacnet_read_args_t Struct Reference

bacnet read property struct

```
#include <handle_property.h>
```

Data Fields

- unsigned int device_id
- int object_numbers
- int object_type [BACNET_READ_ARGS_OBJECT_MAX]
- int object_instance [BACNET_READ_ARGS_OBJECT_MAX]
- int object_property [BACNET_READ_ARGS_OBJECT_MAX]

6.4.1 Detailed Description

bacnet read property struct

Definition at line 57 of file handle_property.h.

6.4.2 Field Documentation

6.4.2.1 unsigned int device_id

device id, This device id is BACnet instance id

Definition at line 60 of file handle_property.h.

6.4.2.2 int object_instance[BACNET_READ_ARGS_OBJECT_MAX]

read object property instance

Definition at line 66 of file handle_property.h.

6.4.2.3 int object_numbers

read object property numbers

Definition at line 62 of file handle_property.h.

6.4.2.4 int object_property[BACNET_READ_ARGS_OBJECT_MAX]

read object property

Definition at line 68 of file handle_property.h.

6.4.2.5 int object_type[BACNET_READ_ARGS_OBJECT_MAX]

read object property type array

Definition at line 64 of file handle_property.h.

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

- include/protocol/bacnet/handle_property.h

6.5 bacnet_write_args_t Struct Reference

bacnet write arg struct

```
#include <handle_property.h>
```

Data Fields

- unsigned int device_id
- int object_type
- int object_instance
- int object_property
- int object_property_priority
- unsigned int object_property_index
- int object_property_value_type
- char object_property_value [32]

6.5.1 Detailed Description

bacnet write arg struct

Definition at line 30 of file handle_property.h.

6.5.2 Field Documentation

6.5.2.1 unsigned int device_id

device id, This device id is BACnet instance id

Definition at line 33 of file handle_property.h.

6.5.2.2 int object_instance

bacnet object instance

Definition at line 37 of file handle_property.h.

6.5.2.3 int object_property

bacnet object property

Definition at line 39 of file handle_property.h.

6.5.2.4 unsigned int object_property_index

bacnet property index, have no index is -1

Definition at line 43 of file handle_property.h.

6.5.2.5 int object_property_priority

bacnet property priority default is 16

Definition at line 41 of file handle_property.h.

6.5.2.6 char object_property_value[32]

bacnet property value

Definition at line 47 of file handle_property.h.

6.5.2.7 int object_property_value_type

bacnet property value type

Definition at line 45 of file handle_property.h.

6.5.2.8 int object_type

bacnet object type

Definition at line 35 of file handle_property.h.

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

- include/protocol/bacnet/handle_property.h

6.6 client_t Struct Reference

Data Fields

- int fd
- uid_t uid

6.6.1 Detailed Description

Definition at line 58 of file service.c.

6.6.2 Field Documentation

6.6.2.1 int fd

Definition at line 60 of file service.c.

6.6.2.2 uid_t uid

Definition at line 61 of file service.c.

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

- src/service.c

6.7 commonBacObj_s Struct Reference

```
#include <device_client.h>
```

Data Fields

- BACNET_OBJECT_TYPE mObject_Type
- uint32_t Object_Instance_Number
- char Object_Name [MAX_DEV_NAME_LEN]

6.7.1 Detailed Description

Structure to define the Object Properties common to all Objects.

Definition at line 177 of file device_client.h.

6.7.2 Field Documentation

6.7.2.1 BACNET_OBJECT_TYPE mObject_Type

The BACnet type of this object (ie, what class is this object from?). This property, of type BACnetObjectType, indicates membership in a particular object type class. Each inherited class will be of one type.

Definition at line 183 of file device_client.h.

6.7.2.2 uint32_t Object_Instance_Number

The instance number for this class instance.

Definition at line 186 of file device_client.h.

6.7.2.3 char Object_Name[MAX_DEV_NAME_LEN]

Object Name; must be unique. This property, of type CharacterString, shall represent a name for the object that is unique within the BACnet Device that maintains it.

Definition at line 192 of file device_client.h.

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

- include/protocol/bacnet/device_client.h

6.8 create_object_return_t Struct Reference

message create a rs485 object return

```
#include <adapter.h>
```

Data Fields

- int object_id

6.8.1 Detailed Description

message create a rs485 object return

Definition at line 74 of file adapter.h.

6.8.2 Field Documentation

6.8.2.1 int object_id

The object ID, the id have created by server

Definition at line 77 of file adapter.h.

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

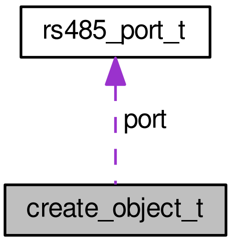
- include/adapter.h

6.9 create_object_t Struct Reference

message create a rs485 object

`#include <adapter.h>`

Collaboration diagram for create_object_t:



Data Fields

- char object_name [36]
- rs485_protocol_type_enum object_type
- int mount_device_max
- rs485_port_t port
- unsigned char address [4]
- int address_len

6.9.1 Detailed Description

message create a rs485 object

Definition at line 52 of file adapter.h.

6.9.2 Field Documentation

6.9.2.1 unsigned char address[4]

The rs485 address, just master device address

Definition at line 63 of file adapter.h.

6.9.2.2 int address_len

The address length

Definition at line 65 of file adapter.h.

6.9.2.3 int mount_device_max

The rs485 object mount max device numbers

Definition at line 59 of file adapter.h.

6.9.2.4 char object_name[36]

The rs485 object name

Definition at line 55 of file adapter.h.

6.9.2.5 rs485_protocol_type_enum object_type

The rs485 protocol type , every protocol type represent a object

Definition at line 57 of file adapter.h.

6.9.2.6 rs485_port_t port

The rs485 port message

Definition at line 61 of file adapter.h.

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

- include/adapter.h

6.10 curtain_profile_t Struct Reference

The curtain profile.

`#include <adapter.h>`

Data Fields

- int current_percent

6.10.1 Detailed Description

The curtain profile.

Definition at line 232 of file adapter.h.

6.10.2 Field Documentation

6.10.2.1 int current_percent

Definition at line 234 of file adapter.h.

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

- include/adapter.h

6.11 delete_object_return_t Struct Reference

message delete a rs485 object return

```
#include <adapter.h>
```

Data Fields

- int delete_status

6.11.1 Detailed Description

message delete a rs485 object return

Definition at line 97 of file adapter.h.

6.11.2 Field Documentation

6.11.2.1 int delete_status

Definition at line 99 of file adapter.h.

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

- include/adapter.h

6.12 delete_object_t Struct Reference

message delete a rs485 object

```
#include <adapter.h>
```

Data Fields

- int object_id

6.12.1 Detailed Description

message delete a rs485 object

Definition at line 86 of file adapter.h.

6.12.2 Field Documentation

6.12.2.1 int object_id

The object id, your want to delete it.

Definition at line 89 of file adapter.h.

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

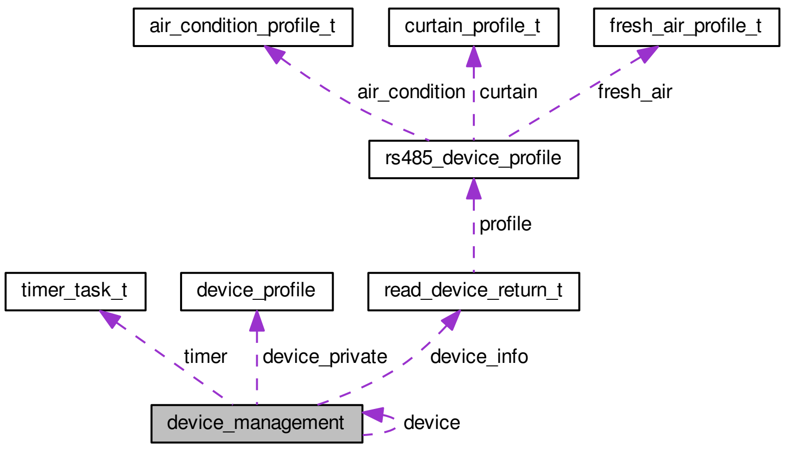
- include/adapter.h

6.13 device_management Struct Reference

device define the device management struct

```
#include <device.h>
```

Collaboration diagram for device_management:



Data Fields

- struct device_management ∗ device
- char ∗ device_name
- rs485_factory_name_enum factory_name
- int object_id
- rs485_protocol_type_enum object_type
- int device_id
- rs485_device_type_enum device_type
- int device_addr_len
- unsigned char device_addr [4]
- unsigned int time_out
- unsigned int device_status_period
- unsigned int retransmission
- bool support_reply
- read_device_return_t ∗ device_info

- timer_task_t ∗ timer
- struct device_profile ∗ device_private
- int device_private_numbers

6.13.1 Detailed Description

device define the device management struct

Definition at line 38 of file device.h.

6.13.2 Field Documentation

6.13.2.1 struct device_management∗ device

The device pointer self

Definition at line 41 of file device.h.

6.13.2.2 unsigned char device_addr[4]

The device address, The used length is 4, just for struct have align

Definition at line 57 of file device.h.

6.13.2.3 int device_addr_len

The device address len

Definition at line 55 of file device.h.

6.13.2.4 int device_id

The device id, It's a key

Definition at line 51 of file device.h.

6.13.2.5 read_device_return_t∗ device_info

The device information

Definition at line 67 of file device.h.

6.13.2.6 char∗ device_name

The device name

Definition at line 43 of file device.h.

6.13.2.7 struct device_profile∗ device_private

The device have a private profile

Definition at line 71 of file device.h.

6.13.2.8 int device_private_numbers

The device private profile numbers

Definition at line 73 of file device.h.

6.13.2.9 unsigned int device_status_period

The rs485 device timer task cyc period

Definition at line 61 of file device.h.

6.13.2.10 rs485_device_type_enum device_type

The device type

Definition at line 53 of file device.h.

6.13.2.11 rs485_factory_name_enum factory_name

The device factory name

Definition at line 45 of file device.h.

6.13.2.12 int object_id

The device belong to RS485 object

Definition at line 47 of file device.h.

6.13.2.13 rs485_protocol_type_enum object_type

The device protocol, It's define by enum.h too

Definition at line 49 of file device.h.

6.13.2.14 unsigned int retransmission

The rs485 device send to fail, and retransmission count

Definition at line 63 of file device.h.

6.13.2.15 bool support_reply

The device have support reply

Definition at line 65 of file device.h.

6.13.2.16 unsigned int time_out

The rs485 device have use the bus time, The max time is 1s

Definition at line 59 of file device.h.

6.13.2.17 timer_task_t∗ timer

The timer task, every device have create a timer task

Definition at line 69 of file device.h.

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

- include/device.h

6.14 device_profile Struct Reference

device_profile device process method

```
#include <support.h>
```

Data Fields

- int addr_real_len
- int method
- method_send send
- method_recv recv

6.14.1 Detailed Description

device_profile device process method

Definition at line 84 of file support.h.

6.14.2 Field Documentation

6.14.2.1 int addr_real_len

Definition at line 87 of file support.h.

6.14.2.2 int method

Definition at line 89 of file support.h.

6.14.2.3 method_recv recv

Definition at line 93 of file support.h.

6.14.2.4 method_send send

Definition at line 91 of file support.h.

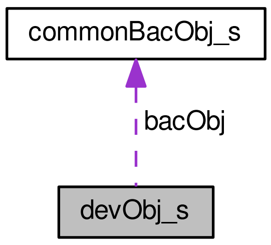
The documentation for this struct was generated from the following file:

- include/support.h

6.15 devObj_s Struct Reference

```
#include <device_client.h>
```

Collaboration diagram for devObj_s:



Data Fields

- BACNET_ADDRESS bacDevAddr
- COMMON_BAC_OBJECT bacObj
- char Description [MAX_DEV_DESC_LEN]
- uint32_t Database_Revision

6.15.1 Detailed Description

Structure to define the Properties of Device Objects which distinguish one instance from another. This structure only defines fields for properties that are unique to a given Device object. The rest may be fixed in device.c or hard-coded into the read-property encoding. This may be useful for implementations which manage multiple Devices, eg, a Gateway.

Definition at line 205 of file device_client.h.

6.15.2 Field Documentation

6.15.2.1 BACNET_ADDRESS bacDevAddr

The BACnet Device Address for this device; ->len depends on DLL type.

Definition at line 207 of file device_client.h.

6.15.2.2 COMMON_BAC_OBJECT bacObj

Structure for the Object Properties common to all Objects.

Definition at line 210 of file device_client.h.

6.15.2.3 uint32_t Database_Revision

The upcounter that shows if the Device ID or object structure has changed.

Definition at line 216 of file device_client.h.

6.15.2.4 char Description[MAX_DEV_DESC_LEN]

Device Description.

Definition at line 213 of file device_client.h.

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

- include/protocol/bacnet/device_client.h

6.16 fresh_air_profile_t Struct Reference

The fresh profile.

```
#include <adapter.h>
```

Data Fields

- int room_temperature
- int room_humidity
- int pm2_5
- int fresh_level

6.16.1 Detailed Description

The fresh profile.

Definition at line 243 of file adapter.h.

6.16.2 Field Documentation

6.16.2.1 int fresh_level

Definition at line 248 of file adapter.h.

6.16.2.2 int pm2_5

Definition at line 247 of file adapter.h.

6.16.2.3 int room_humidity

Definition at line 246 of file adapter.h.

6.16.2.4 int room_temperature

Definition at line 245 of file adapter.h.

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

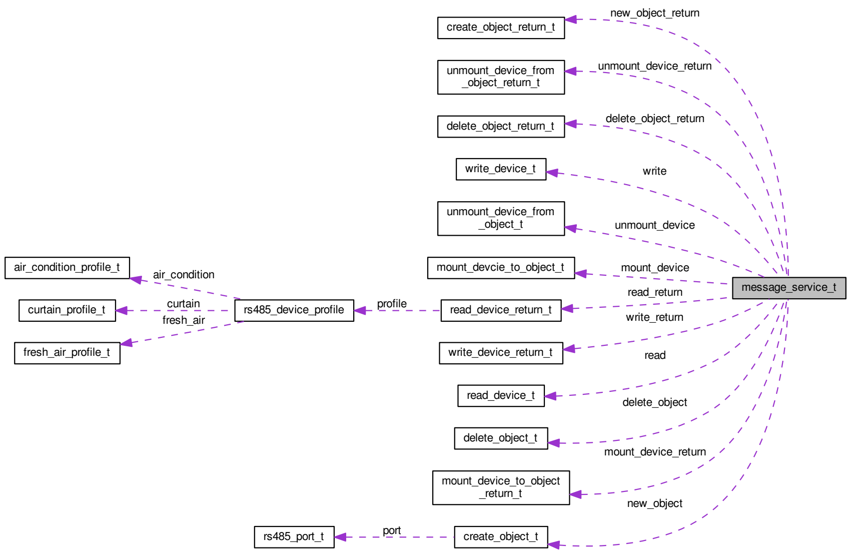
- include/adapter.h

6.17 message_service_t Union Reference

define the receive the message type

`#include <adapter.h>`

Collaboration diagram for message_service_t:



Data Fields

- create_object_t new_object
- delete_object_t delete_object
- mount_devcie_to_object_t mount_device
- unmount_device_from_object_t unmount_device
- write_device_t write
- read_device_t read
- create_object_return_t new_object_return
- delete_object_return_t delete_object_return
- mount_device_to_object_return_t mount_device_return
- unmount_device_from_object_return_t unmount_device_return
- write_device_return_t write_return
- read_device_return_t read_return

6.17.1 Detailed Description

define the receive the message type

Definition at line 289 of file adapter.h.

6.17.2 Field Documentation

6.17.2.1 delete_object_t delete_object

Definition at line 293 of file adapter.h.

6.17.2.2 delete_object_return_t delete_object_return

Definition at line 301 of file adapter.h.

6.17.2.3 mount_devcie_to_object_t mount_device

Definition at line 294 of file adapter.h.

6.17.2.4 mount_device_to_object_return_t mount_device_return

Definition at line 302 of file adapter.h.

6.17.2.5 create_object_t new_object

Definition at line 292 of file adapter.h.

6.17.2.6 create_object_return_t new_object_return

Definition at line 300 of file adapter.h.

6.17.2.7 read_device_t read

Definition at line 297 of file adapter.h.

6.17.2.8 read_device_return_t read_return

Definition at line 305 of file adapter.h.

6.17.2.9 unmount_device_from_object_t unmount_device

Definition at line 295 of file adapter.h.

6.17.2.10 unmount_device_from_object_return_t unmount_device_return

Definition at line 303 of file adapter.h.

6.17.2.11 write_device_t write

Definition at line 296 of file adapter.h.

6.17.2.12 write_device_return_t write_return

Definition at line 304 of file adapter.h.

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

- include/adapter.h

6.18 modbus_port_handle_t Struct Reference

The modbus port interface.

```
#include <modbus.h>
```

Data Fields

- int device_id
- bool broadcast
- unsigned int retransmission
- modbus_function_code_enum code
- int method
- int value
- unsigned char ∗ buffer
- unsigned int buffer_len
- unsigned int device_addr
- unsigned int register_addr
- method_send send_handle
- method_recv recv_handle

6.18.1 Detailed Description

The modbus port interface.

Definition at line 45 of file modbus.h.

6.18.2 Field Documentation

6.18.2.1 bool broadcast

The message data is brodadcase ?

Definition at line 50 of file modbus.h.

6.18.2.2 unsigned char∗ buffer

temp save the data

Definition at line 60 of file modbus.h.

6.18.2.3 unsigned int buffer_len

The buffer length

Definition at line 62 of file modbus.h.

6.18.2.4 modbus_function_code_enum code

The modbus function code, define by enum.h

Definition at line 54 of file modbus.h.

6.18.2.5 unsigned int device_addr

The device address

Definition at line 64 of file modbus.h.

6.18.2.6 int device_id

device id

Definition at line 48 of file modbus.h.

6.18.2.7 int method

The device method,(command) define by enum.h

Definition at line 56 of file modbus.h.

6.18.2.8 method_recv recv_handle

The receive package callback function

Definition at line 70 of file modbus.h.

6.18.2.9 unsigned int register_addr

The you need to operator device register address

Definition at line 66 of file modbus.h.

6.18.2.10 unsigned int retransmission

The device have send fail. retransmission count

Definition at line 52 of file modbus.h.

6.18.2.11 method_send send_handle

The send package callback function

Definition at line 68 of file modbus.h.

6.18.2.12 int value

The method have a value.

Definition at line 58 of file modbus.h.

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

- include/protocol/modbus/modbus.h

6.19 mount_devcie_to_object_t Struct Reference

message mount a device to rs485 object

`#include <adapter.h>`

Data Fields

- char device_name [36]
- rs485_factory_name_enum factory_name
- int object_id
- rs485_protocol_type_enum object_type
- char device_addr [4]
- unsigned int device_addr_len
- rs485_device_type_enum device_type
- unsigned int time_out
- unsigned int support_reply
- unsigned int device_status_period
- unsigned int retransmission

6.19.1 Detailed Description

message mount a device to rs485 object

Definition at line 107 of file adapter.h.

6.19.2 Field Documentation

6.19.2.1 char device_addr[4]

The rs485 device address, and the address maybe to NULL

Definition at line 118 of file adapter.h.

6.19.2.2 unsigned int device_addr_len

The rs485 device address length

Definition at line 120 of file adapter.h.

6.19.2.3 char device_name[36]

The device name

Definition at line 110 of file adapter.h.

6.19.2.4 unsigned int device_status_period

The rs485 device timer task cyc period

Definition at line 128 of file adapter.h.

6.19.2.5 rs485_device_type_enum device_type

The rs485 device type, reference enum.h

Definition at line 122 of file adapter.h.

6.19.2.6 rs485_factory_name_enum factory_name

The device factory

Definition at line 112 of file adapter.h.

6.19.2.7 int object_id

The device mount the which object,so ,you must have crate a object frist

Definition at line 114 of file adapter.h.

6.19.2.8 rs485_protocol_type_enum object_type

The rs485 protocol type, The object type, we need to check it

Definition at line 116 of file adapter.h.

6.19.2.9 unsigned int retransmission

The rs485 device send to fail, and retransmission count

Definition at line 130 of file adapter.h.

6.19.2.10 unsigned int support_reply

The rs485 device have wiat the device reply

Definition at line 126 of file adapter.h.

6.19.2.11 unsigned int time_out

The rs485 device have use the bus time, The max time is 1s

Definition at line 124 of file adapter.h.

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

- include/adapter.h

6.20 mount_device_to_object_return_t Struct Reference

message mount a device to rs485 object return

`#include <adapter.h>`

Data Fields

- int device_id

6.20.1 Detailed Description

message mount a device to rs485 object return

Definition at line 138 of file adapter.h.

6.20.2 Field Documentation

6.20.2.1 int device_id

return the device id , if the device have a negative value, It's mount fail

Definition at line 141 of file adapter.h.

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

- include/adapter.h

6.21 mstp_port_handle Struct Reference

mstp_port_handle general protocol(user defined)

```
#include <general.h>
```

Data Fields

- unsigned char ∗ package_buffer
- unsigned int package_buffer_len
- bool except_reply
- unsigned int retransmission
- unsigned int timeout_ms
- unsigned char address [4]
- unsigned int address_len
- bool broadcast
- method_send send_handle
- int method
- int value
- method_recv recv_handle
- int device_id
- void ∗ arg

6.21.1 Detailed Description

mstp_port_handle general protocol(user defined)

Definition at line 43 of file general.h.

6.21.2 Field Documentation

6.21.2.1 unsigned char address[4]

The device addresss

Definition at line 56 of file general.h.

6.21.2.2 unsigned int address_len

The device address len, should less than 4 byte

Definition at line 58 of file general.h.

6.21.2.3 void∗ arg

resaved argument

Definition at line 72 of file general.h.

6.21.2.4 bool broadcast

The send package is a broadcast ?

Definition at line 60 of file general.h.

6.21.2.5 int device_id

The device id.

Definition at line 70 of file general.h.

6.21.2.6 bool except_reply

Is wait device reply data?

Definition at line 50 of file general.h.

6.21.2.7 int method

The device method

Definition at line 64 of file general.h.

6.21.2.8 unsigned char* package_buffer

The send buffer data to bus

Definition at line 46 of file general.h.

6.21.2.9 unsigned int package_buffer_len

The send buffer length

Definition at line 48 of file general.h.

6.21.2.10 method_recv recv_handle

process the receive buffer

Definition at line 68 of file general.h.

6.21.2.11 unsigned int retransmission

send the data to device have fail, you can retransmission count

Definition at line 52 of file general.h.

6.21.2.12 method_send send_handle

package a send data callback function

Definition at line 62 of file general.h.

6.21.2.13 unsigned int timeout_ms

wait the device reply timeout (ms)

Definition at line 54 of file general.h.

6.21.2.14 int value

The device method include value

Definition at line 66 of file general.h.

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

- include/protocol/general/general.h

6.22 object_functions Struct Reference

```
#include <device_client.h>
```

Data Fields

- BACNET_OBJECT_TYPE Object_Type
- object_init_function Object_Init
- object_count_function Object_Count
- object_index_to_instance_function Object_Index_To_Instance
- object_valid_instance_function Object_Valid_Instance
- object_name_function Object_Name
- read_property_function Object_Read_Property
- write_property_function Object_Write_Property
- rpm_property_lists_function Object_RPM_List
- rr_info_function Object_RR_Info
- object_iterate_function Object_Iterator
- object_value_list_function Object_Value_List
- object_cov_function Object_COV
- object_cov_clear_function Object_COV_Clear
- object_intrinsic_reporting_function Object_Intrinsic_Reporting

6.22.1 Detailed Description

Defines the group of object helper functions for any supported Object.

Each Object must provide some implementation of each of these helpers in order to properly support the handlers. Eg, the ReadProperty handler handler_read_property() relies on the instance of Object_Read_Property for each Object type, or configure the function as NULL. In both appearance and operation, this group of functions acts like they are member functions of a C++ Object base class.

Definition at line 151 of file device_client.h.

6.22.2 Field Documentation

6.22.2.1 object_count_function Object_Count

Definition at line 154 of file device_client.h.

6.22.2.2 object_cov_function Object_COV

Definition at line 164 of file device_client.h.

6.22.2.3 object_cov_clear_function Object_COV_Clear

Definition at line 165 of file device_client.h.

6.22.2.4 object_index_to_instance_function Object_Index_To_Instance

Definition at line 155 of file device_client.h.

6.22.2.5 object_init_function Object_Init

Definition at line 153 of file device_client.h.

6.22.2.6 object_intrinsic_reporting_function Object_Intrinsic_Reporting

Definition at line 166 of file device_client.h.

6.22.2.7 object_iterate_function Object_Iterator

Definition at line 162 of file device_client.h.

6.22.2.8 object_name_function Object_Name

Definition at line 157 of file device_client.h.

6.22.2.9 read_property_function Object_Read_Property

Definition at line 158 of file device_client.h.

6.22.2.10 rpm_property_lists_function Object_RPM_List

Definition at line 160 of file device_client.h.

6.22.2.11 rr_info_function Object_RR_Info

Definition at line 161 of file device_client.h.

6.22.2.12 BACNET_OBJECT_TYPE Object_Type

Definition at line 152 of file device_client.h.

6.22.2.13 object_valid_instance_function Object_Valid_Instance

Definition at line 156 of file device_client.h.

6.22.2.14 object_value_list_function Object_Value_List

Definition at line 163 of file device_client.h.

6.22.2.15 write_property_function Object_Write_Property

Definition at line 159 of file device_client.h.

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

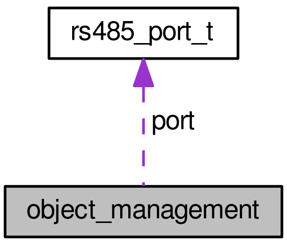
- include/protocol/bacnet/device_client.h

6.23 object_management Struct Reference

object_management define the object management struct

`#include <object.h>`

Collaboration diagram for object_management:



Data Fields

- pthread_t object_thread
- int queue_depth
- int object_id
- char ∗ object_name
- rs485_protocol_type_enum object_type
- int mount_device_max
- rs485_port_t port
- unsigned char address [4]
- int address_len
- int ∗ mount_device
- struct ring_buffer_t ∗ work_queue
- uint8_t ∗ work_queue_buffer
- sem_t queue_sem
- void ∗ object_private

6.23.1 Detailed Description

object_management define the object management struct

Definition at line 56 of file object.h.

6.23.2 Field Documentation

6.23.2.1 unsigned char address[4]

The rs485 object MAC address, It's have 1 byte

Definition at line 73 of file object.h.

6.23.2.2 int address_len

The rs485 object address length

Definition at line 75 of file object.h.

6.23.2.3 int∗ mount_device

The buffer have save the device id, It's malloc

Definition at line 77 of file object.h.

6.23.2.4 int mount_device_max

The rs485 object have mount max device numbers

Definition at line 69 of file object.h.

6.23.2.5 int object_id

The object ID

Definition at line 63 of file object.h.

6.23.2.6 char∗ object_name

The object name , It's malloc

Definition at line 65 of file object.h.

6.23.2.7 void∗ object_private

The pointer have save the device private profile

Definition at line 85 of file object.h.

6.23.2.8 pthread_t object_thread

The linux thread descriptor , it's be used to save the work thread

Definition at line 59 of file object.h.

6.23.2.9 rs485_protocol_type_enum object_type

The rs485 protocol type, It's defined by enum.h

Definition at line 67 of file object.h.

6.23.2.10 rs485_port_t port

The rs485 object UART physics information, The port struct have define by adapter.h

Definition at line 71 of file object.h.

6.23.2.11 int queue_depth

The work thread queue depth

Definition at line 61 of file object.h.

6.23.2.12 sem_t queue_sem

The work queue semaphore , it's be used to save the work queue semaphore

Definition at line 83 of file object.h.

6.23.2.13 struct ring_buffer_t∗ work_queue

The work thread have use a queue

Definition at line 79 of file object.h.

6.23.2.14 uint8_t∗ work_queue_buffer

The work queue buffer, It's malloc

Definition at line 81 of file object.h.

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

- include/object.h

6.24 package Struct Reference

Data Fields

- unsigned char addr_low
- unsigned char addr_high
- unsigned char command
- unsigned char data_addr
- unsigned char data [4]
- int cmd

6.24.1 Detailed Description

Definition at line 77 of file doya.c.

6.24.2 Field Documentation

6.24.2.1 unsigned char addr_high

Definition at line 81 of file doya.c.

6.24.2.2 unsigned char addr_low

Definition at line 80 of file doya.c.

6.24.2.3 int cmd

Definition at line 87 of file doya.c.

6.24.2.4 unsigned char command

Definition at line 82 of file doya.c.

6.24.2.5 unsigned char data[4]

Definition at line 84 of file doya.c.

6.24.2.6 unsigned char data_addr

Definition at line 83 of file doya.c.

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

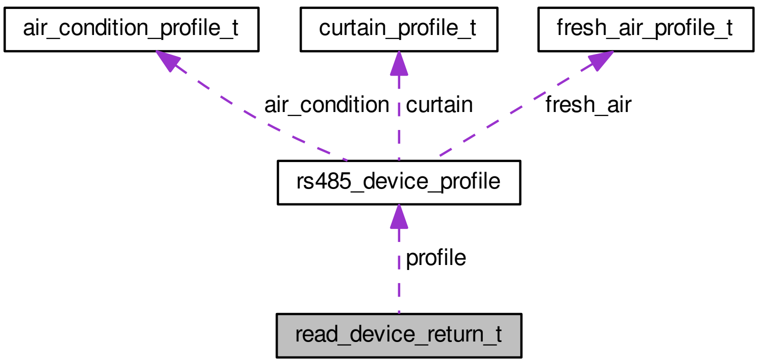
- src/device/curtain/doya/doya.c

6.25 read_device_return_t Struct Reference

message read value from device return

```
#include <adapter.h>
```

Collaboration diagram for read_device_return_t:



Data Fields

- bool read_status
- bool runing
- bool error
- union rs485_device_profile profile

6.25.1 Detailed Description

message read value from device return

Definition at line 271 of file adapter.h.

6.25.2 Field Documentation

6.25.2.1 bool error

the device have a error status

Definition at line 278 of file adapter.h.

6.25.2.2 union **rs485_device_profile** profile

the device profile , have fill it

Definition at line 280 of file adapter.h.

6.25.2.3 bool read_status

The read request status

Definition at line 274 of file adapter.h.

6.25.2.4 bool runing

the read device status

Definition at line 276 of file adapter.h.

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

- include/adapter.h

6.26 read_device_t Struct Reference

message read value from device

```
#include <adapter.h>
```

Data Fields

- int device_id

6.26.1 Detailed Description

message read value from device

Definition at line 203 of file adapter.h.

6.26.2 Field Documentation

6.26.2.1 int device_id

The device id what you want to read device value

Definition at line 206 of file adapter.h.

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

- include/adapter.h

6.27 rs485_curtain_ao_ke_send_package_t Struct Reference

Data Fields

- unsigned char d1
- unsigned char d2
- unsigned char d3
- unsigned char d4
- unsigned char d5

6.27.1 Detailed Description

Definition at line 84 of file aoke.c.

6.27.2 Field Documentation

6.27.2.1 unsigned char d1

Definition at line 86 of file aoke.c.

6.27.2.2 unsigned char d2

Definition at line 87 of file aoke.c.

6.27.2.3 unsigned char d3

Definition at line 88 of file aoke.c.

6.27.2.4 unsigned char d4

Definition at line 89 of file aoke.c.

6.27.2.5 unsigned char d5

Definition at line 90 of file aoke.c.

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

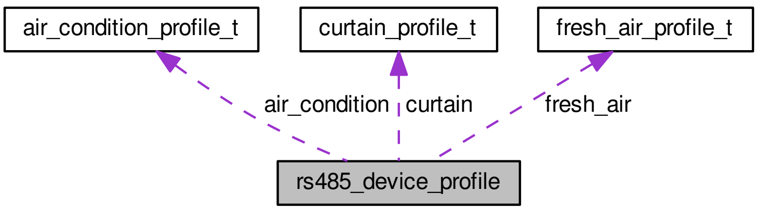
- src/device/curtain/aoke/aoke.c

6.28 rs485_device_profile Union Reference

rs485 device profile

`#include <adapter.h>`

Collaboration diagram for rs485_device_profile:



Data Fields

- air_condition_profile_t air_condition
- curtain_profile_t curtain
- fresh_air_profile_t fresh_air

6.28.1 Detailed Description

rs485 device profile

Definition at line 256 of file adapter.h.

6.28.2 Field Documentation

6.28.2.1 air_condition_profile_t air_condition

The air conditioner profile

Definition at line 259 of file adapter.h.

6.28.2.2 curtain_profile_t curtain

The curtain conditioner profile

Definition at line 261 of file adapter.h.

6.28.2.3 fresh_air_profile_t fresh_air

The fresh air profile

Definition at line 263 of file adapter.h.

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

- include/adapter.h

6.29 rs485_port_t Struct Reference

The rs485 port physical.

```
#include <adapter.h>
```

Data Fields

- unsigned int baud_rate
- char interface_name [16]

6.29.1 Detailed Description

The rs485 port physical.

Definition at line 39 of file adapter.h.

6.29.2 Field Documentation

6.29.2.1 unsigned int baud_rate

The rs485 baud rate, like 9600, 115200 ...

Definition at line 42 of file adapter.h.

6.29.2.2 char interface_name[16]

The rs485 port profile, like /dev/ttyS1, /dev/usbS0....

Definition at line 44 of file adapter.h.

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

- include/adapter.h

6.30 thread_pool_t Struct Reference

define the thread pool struct

```
#include <service.h>
```

Data Fields

- pthread_t ∗ thread
- pthread_attr_t ∗ attr
- void ∗(∗ function)(void ∗arg)
- void ∗ arg
- bool thread_status

6.30.1 Detailed Description

define the thread pool struct

Definition at line 37 of file service.h.

6.30.2 Field Documentation

6.30.2.1 void∗ arg

the thread function argument

Definition at line 46 of file service.h.

6.30.2.2 pthread_attr_t∗ attr

the thread addr argument

Definition at line 42 of file service.h.

6.30.2.3 void∗(∗ function)(void ∗arg)

the thread service function

Definition at line 44 of file service.h.

6.30.2.4 pthread_t∗ thread

the thread Id

Definition at line 40 of file service.h.

6.30.2.5 bool thread_status

the thread create status, It's will used to clean it

Definition at line 48 of file service.h.

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

- include/service.h

6.31 timer_task_t Struct Reference

timer task struct

```
#include <timer_task.h>
```

Data Fields

- unsigned int tick
- unsigned int timeout
- timer_proc_func function
- int device_id
- int command

6.31.1 Detailed Description

timer task struct

Definition at line 39 of file timer_task.h.

6.31.2 Field Documentation

6.31.2.1 int command

The get the device information

Definition at line 50 of file timer_task.h.

6.31.2.2 int device_id

The function argument

Definition at line 48 of file timer_task.h.

6.31.2.3 timer_proc_func function

The timer task function, timeout have call it.

Definition at line 46 of file timer_task.h.

6.31.2.4 unsigned int tick

The timer tick time, sleep every tick

Definition at line 42 of file timer_task.h.

6.31.2.5 unsigned int timeout

The timeout time, when the tick >= timeout, process

Definition at line 44 of file timer_task.h.

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

- include/timer_task.h

6.32 unmount_device_from_object_return_t Struct Reference

message unmount a device from rs485 ojbect return

```
#include <adapter.h>
```

Data Fields

- int unmount_status

6.32.1 Detailed Description

message unmount a device from rs485 ojbect return

Definition at line 162 of file adapter.h.

6.32.2 Field Documentation

6.32.2.1 int unmount_status

The device unmount status

Definition at line 165 of file adapter.h.

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

- include/adapter.h

6.33 unmount_device_from_object_t Struct Reference

message unmount a device form rs485 object

```
#include <adapter.h>
```

Data Fields

- int device_id
- int object_id

6.33.1 Detailed Description

message unmount a device form rs485 object

Definition at line 149 of file adapter.h.

6.33.2 Field Documentation

6.33.2.1 int device_id

The device id what you want to unmount

Definition at line 152 of file adapter.h.

6.33.2.2 int object_id

The object id that the device have mounted

Definition at line 154 of file adapter.h.

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

- include/adapter.h

6.34 write_device_return_t Struct Reference

message write value to device return

`#include <adapter.h>`

Data Fields

- int write_status

6.34.1 Detailed Description

message write value to device return

Definition at line 192 of file adapter.h.

6.34.2 Field Documentation

6.34.2.1 int write_status

The write value return status, just to wirte to work thread have return the struct

Definition at line 195 of file adapter.h.

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

- include/adapter.h

6.35 write_device_t Struct Reference

message write value to device

`#include <adapter.h>`

Data Fields

- int device_id
- bool broadcast
- unsigned int device_method
- int method_value
- int value_reserve

6.35.1 Detailed Description

message write value to device

Definition at line 173 of file adapter.h.

6.35.2 Field Documentation

6.35.2.1 bool broadcast

The write message is broadcast

Definition at line 178 of file adapter.h.

6.35.2.2 int device_id

The device id, you must create a device, you can used it

Definition at line 176 of file adapter.h.

6.35.2.3 unsigned int device_method

The device method, you can reference enum.h

Definition at line 180 of file adapter.h.

6.35.2.4 int method_value

The device method value, just like set the air condition 24

Definition at line 182 of file adapter.h.

6.35.2.5 int value_reserve

The reserve value, you can't used it.

Definition at line 184 of file adapter.h.

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

- include/adapter.h

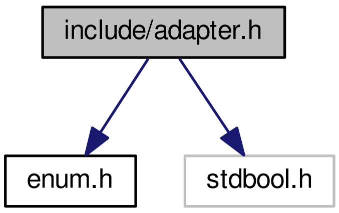
Chapter 7

File Documentation

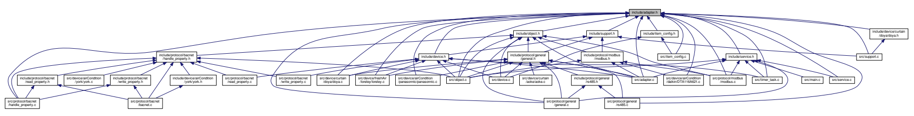
7.1 include/adapter.h File Reference

```
#include "enum.h"
#include <stdbool.h>
```

Include dependency graph for adapter.h:



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



Data Structures

- struct rs485_port_t

  The rs485 port physical.
- struct create_object_t

  message create a rs485 object
- struct create_object_return_t

  message create a rs485 object return
- struct delete_object_t

  message delete a rs485 object

- struct delete_object_return_t

    message delete a rs485 object return
- struct mount_devcie_to_object_t

    message mount a device to rs485 object
- struct mount_device_to_object_return_t

    message mount a device to rs485 object return
- struct unmount_device_from_object_t

    message unmount a device form rs485 object
- struct unmount_device_from_object_return_t

    message unmount a device from rs485 ojbect return
- struct write_device_t

    message write value to device
- struct write_device_return_t

    message write value to device return
- struct read_device_t

    message read value from device
- struct air_condition_profile_t

    The air conditon profile.
- struct curtain_profile_t

    The curtain profile.
- struct fresh_air_profile_t

    The fresh profile.
- union rs485_device_profile

    rs485 device profile
- struct read_device_return_t

    message read value from device return
- union message_service_t

    define the receive the message type
- struct adapter_t

    define the adapter struct

7.1.1 Detailed Description

www.enno.com

Date

: Mar 15, 2016

Author

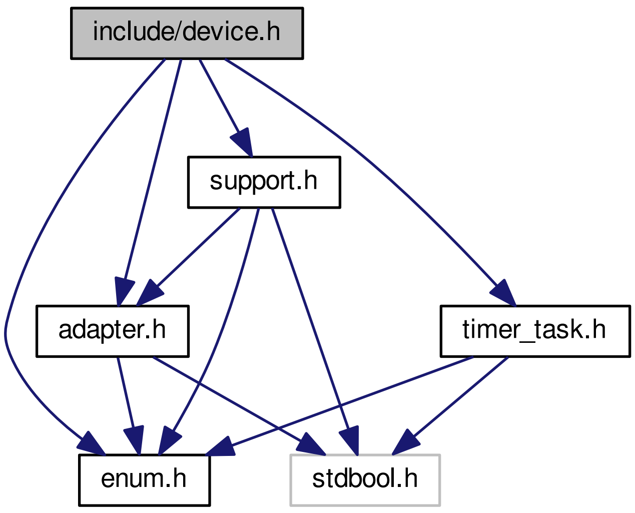
: wong

Definition in file adapter.h.

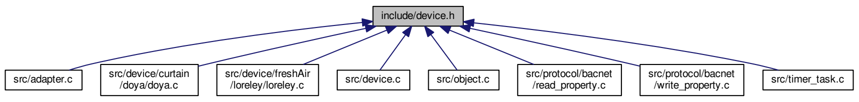
7.2 include/device.h File Reference

```
#include "enum.h"
#include "timer_task.h"
#include "adapter.h"
#include "support.h"
```

Include dependency graph for device.h:



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


include/device.h
src/adapter.c
src/device/curtain /doya/doya.c
src/device/freshAir /loreley/loreley.c
src/device.c
src/object.c
src/protocol/bacnet /read_property.c
src/protocol/bacnet /write_property.c
src/timer_task.c

Data Structures

- struct device_management

  device define the device management struct

Typedefs

- typedef struct device_management device_management_t

  device define the device management struct

Functions

- int create_device (adapter_t ∗adapter) *create_device create a rs485 device, mount the device to protocol*

- int delete_device (int object_id, int device_id)

  delete_device delete a device form device management table.
- int get_device_name (char ∗out, int out_len, int device_id)

  get_device_name get a device name from device database.
- int get_device_type (int device_id)

  get_device_type get a device type from device database, just like air condition, fresh air.....
- int get_device_protocol (int device_id)

  get_device_protocol get a device protocol from device database, just like BACnet, MODUBS...
- int get_device_addr (unsigned char ∗addr, unsigned int addr_len, int device_id)

  get_device_addr get a rs485 device addr, you maybe have no address for some device.
- timer_task_t ∗ get_device_timer (int device_id)

  get_device_timer get a device timer task.
- struct device_profile ∗ get_device_private (int device_id)

  get_device_private get a device private profile
- int get_device_private_numbers (int device_id)

  get_device_private_numbers
- bool check_device_id (int device_id)

  check_object_id check the object is legal
- int get_device_object_id (int device_id)

  get_device_object_id get the object id by device id
- int get_device_factory_name (int device_id)

  get_device_factory_name Get the device factory name
- int get_device_retransmission (int device_id)

  get_device_retransmission Get the device retransmission count on bus
- int get_device_timeout_ms (int device_id)

  get_device_timeout_ms Get The device timeout (ms), The bus have send a package have wait timeout count.
- int get_device_address_len (int device_id)

  get_device_address_len Get the device address length.
- device_management_t ∗ get_device_management (int device_id)

  get_device_management get the device management pointer
- int device_managemnt_init (void)

  device_managemnt_init The device management modele have a initialize
- int set_read_device_information (const read_device_return_t ∗info, int device_id)

  set_read_device_information bus have get a device information have wirte it.
- int get_read_device_information (read_device_return_t ∗out, int device_id)

  get_read_device_information It's read a device information called by adapter layer.

7.2.1 Detailed Description

www.enno.com

Date

: Mar 24, 2016

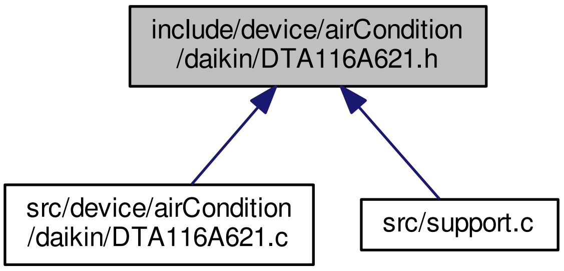
Author

: wong

Definition in file device.h.

7.3 include/device/airCondition/daikin/DTA116A621.h File Reference

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



Functions

- int daikin_dta116a621_set_temperature (volatile void ∗arg)

  daikin_dta116a621_set_temperature set daikin air condition temperature send package to "modbus_port_handle_t"

- int daikin_dta116a621_set_mode (volatile void ∗arg)

  daikin_dta116a621_set_mode set daikin air conditon mode send package to "modbus_port_handle_t"

- int daikin_dta116a621_set_swing (volatile void ∗arg)

  daikin_dta116a621_set_swing set daikin air conditon swing send package to "modbus_port_handle_t"

- int daikin_dta116a621_set_fan (volatile void ∗arg)

  daikin_dta116a621_set_fan set daikin air conditon fan send package to "modbus_port_handle_t"

- int daikin_dta116a621_set_switch (volatile void ∗arg)

  daikin_dta116a621_set_switch set daikin air conditon switch send package to "modbus_port_handle_t"

- int daikin_dta116a621_get_device_info_send (volatile void ∗arg)

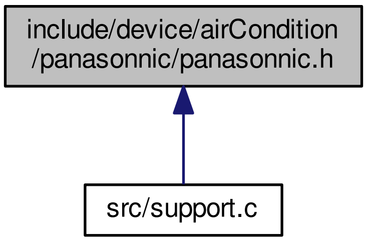
  daikin_dta116a621_get_device_info_send set daikin air conditon device information send package to "modbus_↩ port_handle_t"

- int daikin_dta116a621_get_device_info_handle (volatile void ∗arg)

  daikin_dta116a621_get_device_info_handle process daikin air conditon get device information send package to "modbus_port_handle_t"

7.4 include/device/airCondition/panasonnic/panasonnic.h File Reference

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



Functions

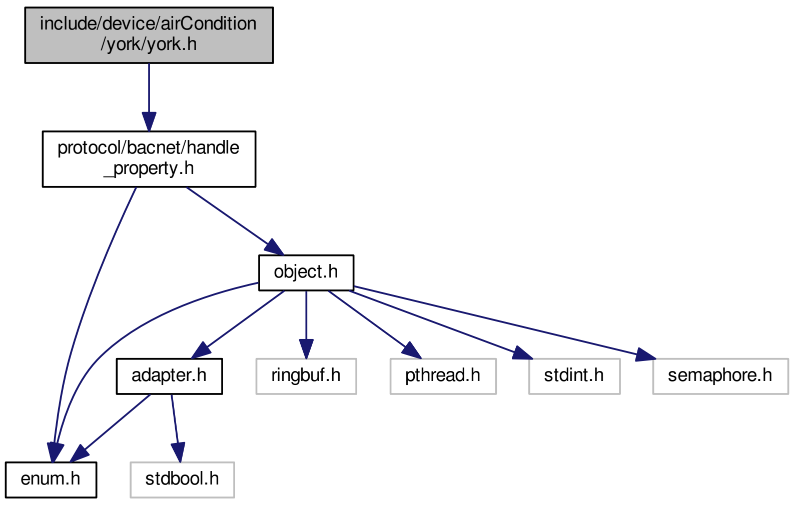
- int panasonnic_send_package_handle (volatile void ∗arg)

  panasonnic_send_package_handle The panasonnic package a send buffer interface.
- int panasonnic_recv_package_handle (volatile void ∗arg)

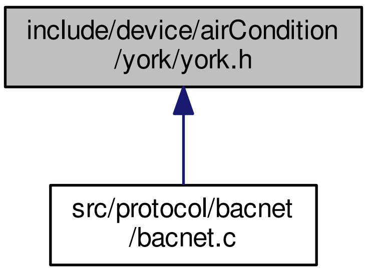
  panasonnic_send_package_handle The panasonnic package a receive buffer processs interface.

7.5 include/device/airCondition/york/york.h File Reference

```
#include "protocol/bacnet/handle_property.h"
```
Include dependency graph for york.h:



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



Functions

- int get_air_york_write_args (bacnet_write_args_t ∗args, unsigned int device_id, int command, int value)

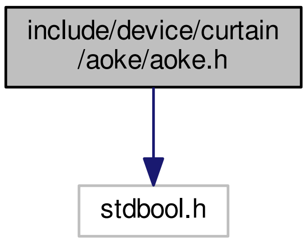
  get_air_york_write_args The york air condition bacnet interface
- int get_air_york_read_args (bacnet_read_args_t ∗args, unsigned int device_id)

  get_air_york_read_args The york air confition bacnet read interface
- int get_air_york_instance (unsigned char mac)

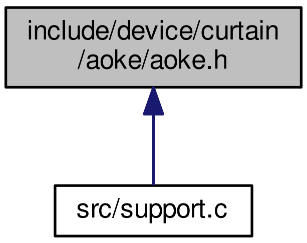
  get_air_york_instance get the youk bacnet instance.

7.6 include/device/curtain/aoke/aoke.h File Reference

`#include <stdbool.h>`
Include dependency graph for aoke.h:



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



Functions

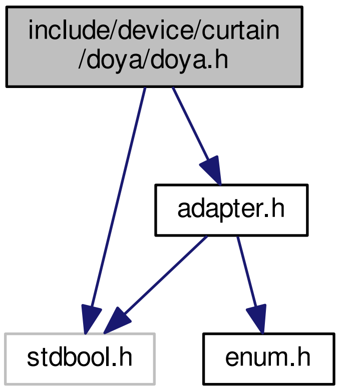
- int aoke_send_package_handle (volatile void ∗arg)

    aoke_send_package_handle aoke curtian package a send buffer
- int aoke_recv_package_handle (volatile void ∗arg)

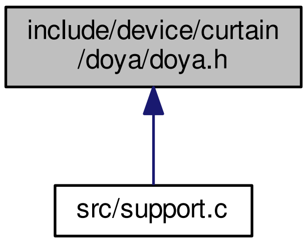
    aoke_recv_package_handle aoke curtain process the receive package

7.7 include/device/curtain/doya/doya.h File Reference

```
#include <stdbool.h>
#include "adapter.h"
```
Include dependency graph for doya.h:



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



Functions

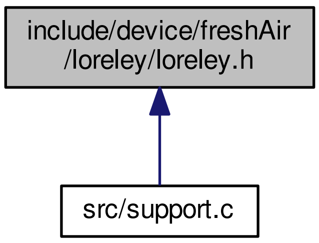
- int doya_send_package_handle (volatile void ∗arg)

  doya_send_package_handle The dooya curtain package a send buffer
- int doya_recv_package_handle (volatile void ∗arg)

  doya_recv_package_handle The dooya curtain process the receive data.

7.8 include/device/freshAir/loreley/loreley.h File Reference

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



Functions

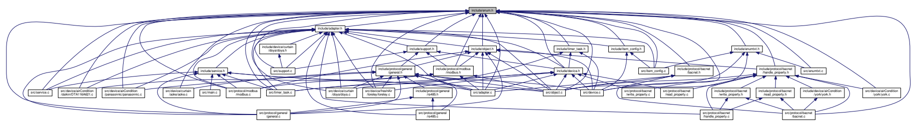
- int loreley_send_package_handle (volatile void ∗arg)

  loreley_send_package_handle loreley fresh air package send a buffer
- int loreley_recv_package_handle (volatile void ∗arg)

  loreley_recv_package_handle loreley fresh air process the receive data.

7.9 include/enum.h File Reference

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



Macros

- #define UNUSED(x) UNUSED_ ## x __attribute__((unused))

Enumerations

- enum timer_task_thread_status_enum {
TIMER_TASK_THREAD_STATUS_START, TIMER_TASK_THREAD_STATUS_INIT, TIMER_TASK_TH↩
READ_STATUS_ADDING, TIMER_TASK_THREAD_STATUS_DELETEING,
TIMER_TASK_THREAD_STATUS_RUNNING, TIMER_TASK_THREAD_STATUS_STOP, TIMER_TASK↩
_THREAD_STATUS_UNKNOWN }

  define the thread status
- enum adapter_thread_status_enum {
ADAPTER_THREAD_STATUS_START, ADAPTER_THREAD_STATUS_INIT, ADAPTER_THREAD_STA↩
TUS_RUNNING, ADAPTER_THREAD_STATUS_STOP,
ADAPTER_THREAD_STATUS_UNKNOWN }
- enum object_thread_status_enum {
OBJECT_THREAD_STATUS_START, OBJECT_THREAD_STATUS_INIT, OBJECT_THREAD_STATUS↩
_RUNNING, OBJECT_THREAD_STATUS_STOP,
OBJECT_THREAD_STATUS_UNKNOWN }
- enum rs485_service_type_enum {
SERVICE_CREATE_RS485_OBJECT, SERVICE_DELETE_RS485_OBJECT, SERVICE_MOUNT_DEVI↩
CE_TO_OBJECT, SERVICE_UNMOUNT_DEVICE_FROM_OBJECT,
SERVICE_WRITE_VALUE_TO_DEVICE, SERVICE_READ_VALUE_FROM_DEVICE, SERVICE_SYSTE↩
M_UPDATE_START, SERVICE_SYSTEM_UPDATE_STOP,
SERVICE_UNKNOWN }

  define the service type
- enum rs485_device_type_enum { RS485_DEVICE_TYPE_AIR_CONDITION, RS485_DEVICE_TYPE_CU↩
RTAIN, RS485_DEVICE_TYPE_FRESH_AIR, RS485_DEVICE_TYPE_UNKNOWN }

  define the device type
- enum rs485_protocol_type_enum { RS485_PROTOCOL_TYPE_BACNET, RS485_PROTOCOL_TYPE_M↩
ODBUS, RS485_PROTOCOL_TYPE_GENERAL, RS485_PROTOCOL_TYPE_UNKNOWN }

  define the protocol type
- enum rs485_factory_name_enum {
RS485_FACTORY_YORK, RS485_FACTORY_PANASONNIC, RS485_FACTORY_DAIKIN_DTA116A621,
RS485_FACTORY_DOYA,
RS485_FACTORY_AOKE, RS485_FACTORY_LORELEY, RS485_FACTORY_UNKNOWN }

  define the support information
- enum modbus_function_code_enum {
MODBUS_FUNCTION_CODE_WRITE_SIGNLE_COIL, MODBUS_FUNCTION_CODE_READ_SIGNLE_↩
COIL, MODBUS_FUNCTION_CODE_WRITE_MULTIPLE_COILS, MODBUS_FUNCTION_CODE_READ↩
_MULTIPLE_COILS,
MODBUS_FUNCTION_CODE_WRITE_SIGNLE_REGISTER, MODBUS_FUNCTION_CODE_READ_SIG↩
NLE_REGISTER, MODBUS_FUNCTION_CODE_WRITE_MULTIPLE_REGISTERS, MODBUS_FUNCTI↩

ON_CODE_READ_MULTIPLE_REGISTERS,
MODBUS_FUNCTION_CODE_DO_NOTHING }

- enum rs485_method_air_condition_york_enum {
RS485_YORK_AIR_SET_TEMP_18 = 18, RS485_YORK_AIR_SET_TEMP_19 = 19, RS485_YORK_AIR↩
_SET_TEMP_20 = 20, RS485_YORK_AIR_SET_TEMP_21 = 21,
RS485_YORK_AIR_SET_TEMP_22 = 22, RS485_YORK_AIR_SET_TEMP_23 = 23, RS485_YORK_AIR↩
_SET_TEMP_24 = 24, RS485_YORK_AIR_SET_TEMP_25 = 25,
RS485_YORK_AIR_SET_TEMP_26 = 26, RS485_YORK_AIR_SET_TEMP_27 = 27, RS485_YORK_AIR↩
_SET_TEMP_28 = 28, RS485_YORK_AIR_SET_TEMP_29 = 29,
RS485_YORK_AIR_SET_TEMP_30 = 30, RS485_YORK_AIR_SET_TEMP_31 = 31, RS485_YORK_AIR↩
_SET_TEMP_32 = 32, RS485_YORK_AIR_SET_HUMIDITY = 33,
RS485_YORK_AIR_SWING_AUTO = 41, RS485_YORK_AIR_SWING_UP_DOWN = 42, RS485_YORK_↩
AIR_SWING_LEFT_RIGHT = 43, RS485_YORK_AIR_SWING_UP_DOWN_LEFT_RIGHT = 44,
RS485_YORK_AIR_FAN_AUTO = 51, RS485_YORK_AIR_FAN_HIGH = 52, RS485_YORK_AIR_FAN_M↩
IDDLE = 53, RS485_YORK_AIR_FAN_LOW = 54,
RS485_YORK_AIR_MODE_FANING = 61, RS485_YORK_AIR_MODE_HEATING = 62, RS485_YORK_A↩
IR_MODE_COOLING = 63, RS485_YORK_AIR_MODE_DRYING = 64,
RS485_YORK_AIR_MODE_AUTOING = 65, RS485_YORK_AIR_OFF = 77, RS485_YORK_AIR_ON = 78,
RS485_YORK_AIR_GET_DEVICE_INFO = 81,
RS485_YORK_AIR_ERR_RESET_YES, RS485_YORK_AIR_ERR_RESET_NO, RS485_YORK_AIR_NE↩
T_RESET_YES, RS485_YORK_AIR_NET_RESET_NO,
RS485_YORK_AIR_SLEEP_YES, RS485_YORK_AIR_SLEEP_NO, RS485_YORK_AIR_ELECTRICAL_↩
HEAT_YES, RS485_YORK_AIR_ELECTRICAL_HEAT_NO,
RS485_YORK_AIR_HEALTH_AIR_YES, RS485_YORK_AIR_HEALTH_AIR_NO, RS485_YORK_AIR_H↩
OT_WATER_YES, RS485_YORK_AIR_HOT_WATER_NO,
RS485_YORK_AIR_HOME_LEFT_YES, RS485_YORK_AIR_HOME_LEFT_NO, RS485_YORK_AIR_FIX↩
_RUN_YES, RS485_YORK_AIR_FIX_RUN_NO,
RS485_YORK_AIR_SAVING_YES, RS485_YORK_AIR_SAVING_NO, RS485_YORK_AIR_DEFROST_Y↩
ES, RS485_YORK_AIR_DEFROST_NO,
RS485_YORK_AIR_COOL_ONLY_YES, RS485_YORK_AIR_COOL_ONLY_NO, RS485_YORK_AIR_CE↩
NTRAL_CONTROL_ONLY_YES, RS485_YORK_AIR_CENTRAL_CONTROL_ONLY_NO }
- enum rs485_method_air_condition_panasonnic_enum {
RS485_PANASONNIC_AIR_SET_TEMP_16 = 16, RS485_PANASONNIC_AIR_SET_TEMP_17 = 17, R↩
S485_PANASONNIC_AIR_SET_TEMP_18 = 18, RS485_PANASONNIC_AIR_SET_TEMP_19 = 19,
RS485_PANASONNIC_AIR_SET_TEMP_20 = 20, RS485_PANASONNIC_AIR_SET_TEMP_21 = 21, R↩
S485_PANASONNIC_AIR_SET_TEMP_22 = 22, RS485_PANASONNIC_AIR_SET_TEMP_23 = 23,
RS485_PANASONNIC_AIR_SET_TEMP_24 = 24, RS485_PANASONNIC_AIR_SET_TEMP_25 = 25, R↩
S485_PANASONNIC_AIR_SET_TEMP_26 = 26, RS485_PANASONNIC_AIR_SET_TEMP_27 = 27,
RS485_PANASONNIC_AIR_SET_TEMP_28 = 28, RS485_PANASONNIC_AIR_SET_TEMP_29 = 29, R↩
S485_PANASONNIC_AIR_SET_TEMP_30 = 30, RS485_PANASONNIC_AIR_SWING_AUTO = 41,
RS485_PANASONNIC_AIR_SWING_HAND5 = 42, RS485_PANASONNIC_AIR_SWING_HAND4 = 43, R↩
S485_PANASONNIC_AIR_SWING_HAND3 = 44, RS485_PANASONNIC_AIR_SWING_HAND2 = 45,
RS485_PANASONNIC_AIR_SWING_HAND1 = 46, RS485_PANASONNIC_AIR_FAN_AUTO = 51, R↩
S485_PANASONNIC_AIR_FAN_HIGH = 52, RS485_PANASONNIC_AIR_FAN_MIDDLE = 53,
RS485_PANASONNIC_AIR_FAN_LOW = 54, RS485_PANASONNIC_AIR_FAN_MOST = 55, RS485_PA↩
NASONNIC_AIR_FAN_MUTE = 56, RS485_PANASONNIC_AIR_MODE_FANING = 61,
RS485_PANASONNIC_AIR_MODE_HEATING = 62, RS485_PANASONNIC_AIR_MODE_COOLING = 63,
RS485_PANASONNIC_AIR_MODE_DRYING = 64, RS485_PANASONNIC_AIR_MODE_AUTOING = 65,
RS485_PANASONNIC_AIR_OFF = 71, RS485_PANASONNIC_AIR_ON = 72, RS485_PANASONNIC_AI↩
R_RESET = 73, RS485_PANASONNIC_AIR_GET_DEVICE_INFO = 81 }
- enum rs485_method_curtain_aoke_enum {
RS485_AOKE_CURTAIN_OPEN = 101, RS485_AOKE_CURTAIN_CLOSE = 102, RS485_AOKE_CURT↩
AIN_SET_PERCENT = 103, RS485_AOKE_CURTAIN_RESET = 104,
RS485_AOKE_CURTAIN_GET_DEVICE_INFO = 105 }
- enum rs485_method_curtain_doya_enum {
RS485_DOYA_CURTAIN_OPEN = 101, RS485_DOYA_CURTAIN_CLOSE = 102, RS485_DOYA_CURT↩
AIN_SET_PERCENT = 103, RS485_DOYA_CURTAIN_RESET = 104,
RS485_DOYA_CURTAIN_GET_DEVICE_INFO = 105 }

- enum rs485_method_fresh_air_loreley_enum { RS485_LORELEY_FRESH_AIR_AUTO_ON = 201, RS485↩ _LORELEY_FRESH_AIR_AUTO_OFF = 202, RS485_LORELEY_FRESH_AIR_RESET = 203, RS485_L↩ ORELEY_FRESH_AIR_GET_DEVICE_INFO = 204 }
- enum rs485_device_method_enum {
  RS485_AIR_SET_TEMP = 10, RS485_AIR_SET_TEMP_18 = 18, RS485_AIR_SET_TEMP_19 = 19, R↩ S485_AIR_SET_TEMP_20 = 20,
  RS485_AIR_SET_TEMP_21 = 21, RS485_AIR_SET_TEMP_22 = 22, RS485_AIR_SET_TEMP_23 = 23, RS485_AIR_SET_TEMP_24 = 24,
  RS485_AIR_SET_TEMP_25 = 25, RS485_AIR_SET_TEMP_26 = 26, RS485_AIR_SET_TEMP_27 = 27, RS485_AIR_SET_TEMP_28 = 28,
  RS485_AIR_SET_TEMP_29 = 29, RS485_AIR_SET_TEMP_30 = 30, RS485_AIR_SWING = 40, RS485_↩ AIR_SWING_AUTO = 41,
  RS485_AIR_SWING_UP_DOWN = 42, RS485_AIR_SWING_LEFT_RIGHT = 43, RS485_AIR_SWING_U↩ P_DOWN_LEFT_RIGHT = 44, RS485_AIR_FAN = 50,
  RS485_AIR_FAN_AUTO = 51, RS485_AIR_FAN_HIGH = 52, RS485_AIR_FAN_MIDDLE = 53, RS485_A↩ IR_FAN_LOW = 54,
  RS485_AIR_MODE = 60, RS485_AIR_MODE_FANING = 61, RS485_AIR_MODE_HEATING = 62, RS485↩ _AIR_MODE_COOLING = 63,
  RS485_AIR_MODE_DRYING = 64, RS485_AIR_MODE_AUTOING = 65, RS485_AIR_SWITCH = 70, R↩ S485_AIR_OFF = 71,
  RS485_AIR_ON = 72, RS485_AIR_RESTART = 73, RS485_AIR_GET_DEVICE_INFO = 81, RS485_CU↩ RTAIN = 100,
  RS485_CURTAIN_OPEN = 101, RS485_CURTAIN_CLOSE = 102, RS485_CURTAIN_SET_PERCENT = 103, RS485_CURTAIN_RESET = 104,
  RS485_CURTAIN_GET_DEVICE_INFO = 105, RS485_FRESH_AIR = 200, RS485_FRESH_AIR_AUTO_↩ ON = 201, RS485_FRESH_AIR_AUTO_OFF = 202,
  RS485_FRESH_AIR_RESET = 203, RS485_FRESH_AIR_GET_DEVICE_INFO = 204 }

7.9.1 Detailed Description

www.enno.com

Date

: Mar 15, 2016

Author

: wong

Definition in file enum.h.

7.9.2 Macro Definition Documentation

7.9.2.1 #define UNUSED(*x*) UNUSED_ ## x __attribute__((unused))

define the unused mac

Definition at line 387 of file enum.h.

7.9.3 Enumeration Type Documentation

7.9.3.1 enum adapter_thread_status_enum

define the adapter thread run status

Enumerator

ADAPTER_THREAD_STATUS_START
ADAPTER_THREAD_STATUS_INIT
ADAPTER_THREAD_STATUS_RUNNING
ADAPTER_THREAD_STATUS_STOP
ADAPTER_THREAD_STATUS_UNKNOWN

Definition at line 38 of file enum.h.

7.9.3.2 enum modbus_function_code_enum

Enumerator

MODBUS_FUNCTION_CODE_WRITE_SIGNLE_COIL
MODBUS_FUNCTION_CODE_READ_SIGNLE_COIL
MODBUS_FUNCTION_CODE_WRITE_MULTIPLE_COILS
MODBUS_FUNCTION_CODE_READ_MULTIPLE_COILS
MODBUS_FUNCTION_CODE_WRITE_SIGNLE_REGISTER
MODBUS_FUNCTION_CODE_READ_SIGNLE_REGISTER
MODBUS_FUNCTION_CODE_WRITE_MULTIPLE_REGISTERS
MODBUS_FUNCTION_CODE_READ_MULTIPLE_REGISTERS
MODBUS_FUNCTION_CODE_DO_NOTHING

Definition at line 136 of file enum.h.

7.9.3.3 enum object_thread_status_enum

define the object thread run status

Enumerator

OBJECT_THREAD_STATUS_START
OBJECT_THREAD_STATUS_INIT
OBJECT_THREAD_STATUS_RUNNING
OBJECT_THREAD_STATUS_STOP
OBJECT_THREAD_STATUS_UNKNOWN

Definition at line 49 of file enum.h.

7.9.3.4 enum rs485_device_method_enum

define the device support method

Enumerator

RS485_AIR_SET_TEMP
RS485_AIR_SET_TEMP_18
RS485_AIR_SET_TEMP_19
RS485_AIR_SET_TEMP_20
RS485_AIR_SET_TEMP_21
RS485_AIR_SET_TEMP_22

RS485_AIR_SET_TEMP_23

RS485_AIR_SET_TEMP_24

RS485_AIR_SET_TEMP_25

RS485_AIR_SET_TEMP_26

RS485_AIR_SET_TEMP_27

RS485_AIR_SET_TEMP_28

RS485_AIR_SET_TEMP_29

RS485_AIR_SET_TEMP_30

RS485_AIR_SWING

RS485_AIR_SWING_AUTO

RS485_AIR_SWING_UP_DOWN

RS485_AIR_SWING_LEFT_RIGHT

RS485_AIR_SWING_UP_DOWN_LEFT_RIGHT

RS485_AIR_FAN

RS485_AIR_FAN_AUTO

RS485_AIR_FAN_HIGH

RS485_AIR_FAN_MIDDLE

RS485_AIR_FAN_LOW

RS485_AIR_MODE

RS485_AIR_MODE_FANING

RS485_AIR_MODE_HEATING

RS485_AIR_MODE_COOLING

RS485_AIR_MODE_DRYING

RS485_AIR_MODE_AUTOING

RS485_AIR_SWITCH

RS485_AIR_OFF

RS485_AIR_ON

RS485_AIR_RESTART

RS485_AIR_GET_DEVICE_INFO

RS485_CURTAIN

RS485_CURTAIN_OPEN

RS485_CURTAIN_CLOSE

RS485_CURTAIN_SET_PERCENT

RS485_CURTAIN_RESET

RS485_CURTAIN_GET_DEVICE_INFO

RS485_FRESH_AIR

RS485_FRESH_AIR_AUTO_ON

RS485_FRESH_AIR_AUTO_OFF

RS485_FRESH_AIR_RESET

RS485_FRESH_AIR_GET_DEVICE_INFO

Definition at line 320 of file enum.h.

7.9.3.5 enum rs485_device_type_enum

define the device type

define the rs485 device type

Enumerator

RS485_DEVICE_TYPE_AIR_CONDITION

RS485_DEVICE_TYPE_CURTAIN

RS485_DEVICE_TYPE_FRESH_AIR

RS485_DEVICE_TYPE_UNKNOWN

Definition at line 91 of file enum.h.

7.9.3.6 enum rs485_factory_name_enum

define the support information

define the factory name

Enumerator

RS485_FACTORY_YORK

RS485_FACTORY_PANASONNIC

RS485_FACTORY_DAIKIN_DTA116A621

RS485_FACTORY_DOYA

RS485_FACTORY_AOKE

RS485_FACTORY_LORELEY

RS485_FACTORY_UNKNOWN

Definition at line 124 of file enum.h.

7.9.3.7 enum rs485_method_air_condition_panasonnic_enum

Enumerator

RS485_PANASONNIC_AIR_SET_TEMP_16

RS485_PANASONNIC_AIR_SET_TEMP_17

RS485_PANASONNIC_AIR_SET_TEMP_18

RS485_PANASONNIC_AIR_SET_TEMP_19

RS485_PANASONNIC_AIR_SET_TEMP_20

RS485_PANASONNIC_AIR_SET_TEMP_21

RS485_PANASONNIC_AIR_SET_TEMP_22

RS485_PANASONNIC_AIR_SET_TEMP_23

RS485_PANASONNIC_AIR_SET_TEMP_24

RS485_PANASONNIC_AIR_SET_TEMP_25

RS485_PANASONNIC_AIR_SET_TEMP_26

RS485_PANASONNIC_AIR_SET_TEMP_27

RS485_PANASONNIC_AIR_SET_TEMP_28

RS485_PANASONNIC_AIR_SET_TEMP_29

RS485_PANASONNIC_AIR_SET_TEMP_30

RS485_PANASONNIC_AIR_SWING_AUTO

RS485_PANASONNIC_AIR_SWING_HAND5

RS485_PANASONNIC_AIR_SWING_HAND4

RS485_PANASONNIC_AIR_SWING_HAND3

RS485_PANASONNIC_AIR_SWING_HAND2

RS485_PANASONNIC_AIR_SWING_HAND1

RS485_PANASONNIC_AIR_FAN_AUTO

RS485_PANASONNIC_AIR_FAN_HIGH

RS485_PANASONNIC_AIR_FAN_MIDDLE

RS485_PANASONNIC_AIR_FAN_LOW

RS485_PANASONNIC_AIR_FAN_MOST

RS485_PANASONNIC_AIR_FAN_MUTE

RS485_PANASONNIC_AIR_MODE_FANING

RS485_PANASONNIC_AIR_MODE_HEATING

RS485_PANASONNIC_AIR_MODE_COOLING

RS485_PANASONNIC_AIR_MODE_DRYING

RS485_PANASONNIC_AIR_MODE_AUTOING

RS485_PANASONNIC_AIR_OFF

RS485_PANASONNIC_AIR_ON

RS485_PANASONNIC_AIR_RESET

RS485_PANASONNIC_AIR_GET_DEVICE_INFO

Definition at line 241 of file enum.h.

7.9.3.8 enum rs485_method_air_condition_york_enum

device method define

Enumerator

RS485_YORK_AIR_SET_TEMP_18

RS485_YORK_AIR_SET_TEMP_19

RS485_YORK_AIR_SET_TEMP_20

RS485_YORK_AIR_SET_TEMP_21

RS485_YORK_AIR_SET_TEMP_22

RS485_YORK_AIR_SET_TEMP_23

RS485_YORK_AIR_SET_TEMP_24

RS485_YORK_AIR_SET_TEMP_25

RS485_YORK_AIR_SET_TEMP_26

RS485_YORK_AIR_SET_TEMP_27

RS485_YORK_AIR_SET_TEMP_28

RS485_YORK_AIR_SET_TEMP_29

RS485_YORK_AIR_SET_TEMP_30

RS485_YORK_AIR_SET_TEMP_31

RS485_YORK_AIR_SET_TEMP_32

RS485_YORK_AIR_SET_HUMIDITY

RS485_YORK_AIR_SWING_AUTO

RS485_YORK_AIR_SWING_UP_DOWN

RS485_YORK_AIR_SWING_LEFT_RIGHT

RS485_YORK_AIR_SWING_UP_DOWN_LEFT_RIGHT

RS485_YORK_AIR_FAN_AUTO

RS485_YORK_AIR_FAN_HIGH

RS485_YORK_AIR_FAN_MIDDLE

RS485_YORK_AIR_FAN_LOW

RS485_YORK_AIR_MODE_FANING

RS485_YORK_AIR_MODE_HEATING

RS485_YORK_AIR_MODE_COOLING

RS485_YORK_AIR_MODE_DRYING

RS485_YORK_AIR_MODE_AUTOING

RS485_YORK_AIR_OFF

RS485_YORK_AIR_ON

RS485_YORK_AIR_GET_DEVICE_INFO

RS485_YORK_AIR_ERR_RESET_YES

RS485_YORK_AIR_ERR_RESET_NO

RS485_YORK_AIR_NET_RESET_YES

RS485_YORK_AIR_NET_RESET_NO

RS485_YORK_AIR_SLEEP_YES

RS485_YORK_AIR_SLEEP_NO

RS485_YORK_AIR_ELECTRICAL_HEAT_YES

RS485_YORK_AIR_ELECTRICAL_HEAT_NO

RS485_YORK_AIR_HEALTH_AIR_YES

RS485_YORK_AIR_HEALTH_AIR_NO

RS485_YORK_AIR_HOT_WATER_YES

RS485_YORK_AIR_HOT_WATER_NO

RS485_YORK_AIR_HOME_LEFT_YES

RS485_YORK_AIR_HOME_LEFT_NO

RS485_YORK_AIR_FIX_RUN_YES

RS485_YORK_AIR_FIX_RUN_NO

RS485_YORK_AIR_SAVING_YES

RS485_YORK_AIR_SAVING_NO

RS485_YORK_AIR_DEFROST_YES

RS485_YORK_AIR_DEFROST_NO

RS485_YORK_AIR_COOL_ONLY_YES

RS485_YORK_AIR_COOL_ONLY_NO

RS485_YORK_AIR_CENTRAL_CONTROL_ONLY_YES

RS485_YORK_AIR_CENTRAL_CONTROL_ONLY_NO

Definition at line 161 of file enum.h.

7.9.3.9 enum rs485_method_curtain_aoke_enum

Enumerator

RS485_AOKE_CURTAIN_OPEN

RS485_AOKE_CURTAIN_CLOSE

RS485_AOKE_CURTAIN_SET_PERCENT

RS485_AOKE_CURTAIN_RESET

RS485_AOKE_CURTAIN_GET_DEVICE_INFO

Definition at line 288 of file enum.h.

7.9.3.10 enum rs485_method_curtain_doya_enum

Enumerator

RS485_DOYA_CURTAIN_OPEN

RS485_DOYA_CURTAIN_CLOSE

RS485_DOYA_CURTAIN_SET_PERCENT

RS485_DOYA_CURTAIN_RESET

RS485_DOYA_CURTAIN_GET_DEVICE_INFO

Definition at line 298 of file enum.h.

7.9.3.11 enum rs485_method_fresh_air_loreley_enum

Enumerator

RS485_LORELEY_FRESH_AIR_AUTO_ON

RS485_LORELEY_FRESH_AIR_AUTO_OFF

RS485_LORELEY_FRESH_AIR_RESET

RS485_LORELEY_FRESH_AIR_GET_DEVICE_INFO

Definition at line 309 of file enum.h.

7.9.3.12 enum rs485_protocol_type_enum

define the protocol type

define the rs485 protocol type

Enumerator

RS485_PROTOCOL_TYPE_BACNET

RS485_PROTOCOL_TYPE_MODBUS

RS485_PROTOCOL_TYPE_GENERAL

RS485_PROTOCOL_TYPE_UNKNOWN

Definition at line 107 of file enum.h.

7.9.3.13 enum rs485_service_type_enum

define the service type

define the adapter message type

Enumerator

SERVICE_CREATE_RS485_OBJECT

SERVICE_DELETE_RS485_OBJECT

SERVICE_MOUNT_DEVICE_TO_OBJECT

SERVICE_UNMOUNT_DEVICE_FROM_OBJECT

SERVICE_WRITE_VALUE_TO_DEVICE

SERVICE_READ_VALUE_FROM_DEVICE

SERVICE_SYSTEM_UPDATE_START

SERVICE_SYSTEM_UPDATE_STOP

SERVICE_UNKNOWN

Definition at line 69 of file enum.h.

7.9.3.14 enum timer_task_thread_status_enum

define the thread status

define the timer task thread run status

Enumerator

TIMER_TASK_THREAD_STATUS_START

TIMER_TASK_THREAD_STATUS_INIT

TIMER_TASK_THREAD_STATUS_ADDING

TIMER_TASK_THREAD_STATUS_DELETEING

TIMER_TASK_THREAD_STATUS_RUNNING

TIMER_TASK_THREAD_STATUS_STOP

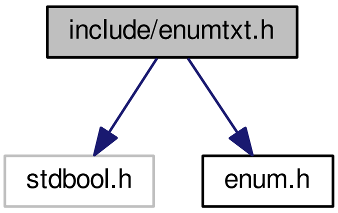
TIMER_TASK_THREAD_STATUS_UNKNOWN

Definition at line 25 of file enum.h.

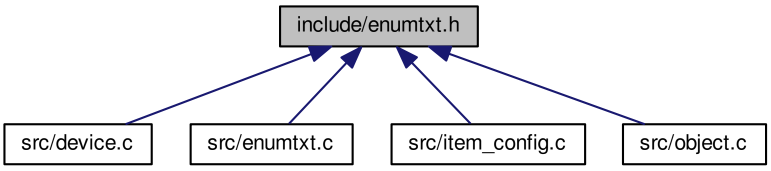
7.10 include/enumtxt.h File Reference

```
#include <stdbool.h>
#include "enum.h"
```

Include dependency graph for enumtxt.h:



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



Functions

- char ∗ get_enum_txt_service (rs485_service_type_enum type)

    get_enum_txt_service get enum rs485 service message type
- char ∗ get_enum_txt_rs485_device_type (rs485_device_type_enum type)

    get_enum_txt_rs485_device_type get enum rs485 device type
- char ∗ get_enum_txt_rs485_protocol_type (rs485_protocol_type_enum type)

    get_enum_txt_rs485_protocol_type get enum rs485 protocol type
- char ∗ get_enum_txt_device_method (rs485_device_method_enum type)

    get_enum_txt_device_method get enum device method(command)
- char ∗ get_enum_txt_device_factory (rs485_factory_name_enum name)

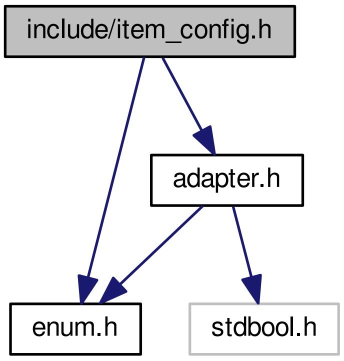
    get_enum_txt_device_factory get enum device factory name
- char ∗ get_enum_txt_bool (bool status)

    get_enum_txt_bool get the string about bool value

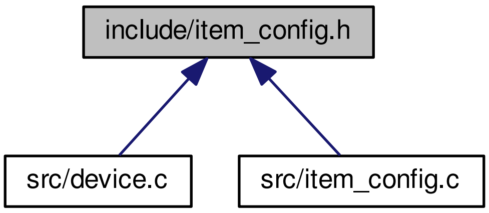
7.11 include/item_config.h File Reference

```
#include "enum.h"
#include "adapter.h"
```

Include dependency graph for item_config.h:



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



Macros

- #define PANNO_S_ITEM_CONFIG
- #define PANNO_S_ITEM_DEFAULT (1)
- #define PANNO_S_ITEM_WENRUDE (0)
- #define PANNO_S_ITEM_ARMANI (0)
- #define PANNO_S_ITEM_SHAOCHENGGUOJI (0)

Functions

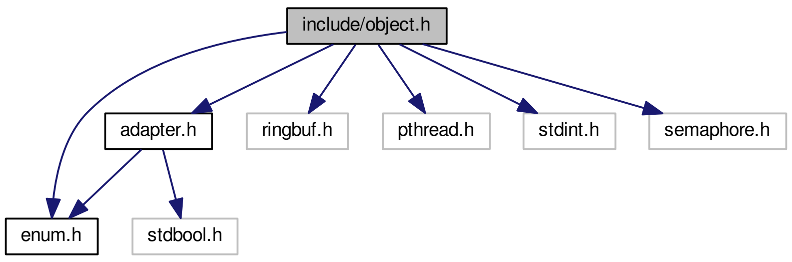
- void panno_s_item_config (adapter_t ∗adapter, rs485_device_type_enum device_type, unsigned char device_addr)

    panno_s_item_config This function is offter the pannoS item config

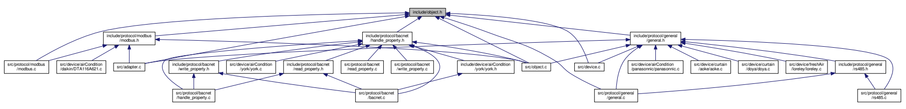
7.12 include/object.h File Reference

```
#include "enum.h"
#include "adapter.h"
#include "ringbuf.h"
#include <pthread.h>
#include <stdint.h>
#include <semaphore.h>
```

Include dependency graph for object.h:



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



Data Structures

- struct object_management

  object_management define the object management struct

Typedefs

- typedef struct object_management object_management_t

  object_management define the object management struct

Functions

- int create_object (const adapter_t ∗adapter)

  create_object create a object by the adapter message
- int delete_object (int object_id)

  delete_object delete a rs485 object by object id
- bool check_object_id (int object_id)

  check_object_id check the object is legal
- int get_object_type (int object_id)

  get_object_type get the object protocol type
- int get_object_mount_device (int object_id, int ∗out_id, int out_id_len)

get_object_mount_device get the object mount device
- bool check_object_numbers_have_idle (int object_id)

  check_object_numbers_have_idle check object mount device is full ?
- int object_mount_device_id (int object_id, int device_id)

  object_mount_device_id add a device to his object
- void object_unmount_device_id (int object_id, int device_id)

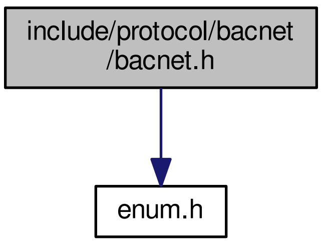
  object_unmount_device_id delete a device form his object
- void ∗ get_object_work_queue (int object_id)

  get_object_work_queue get the object of work queue
- void ∗ get_object_queue_sem (int object_id)

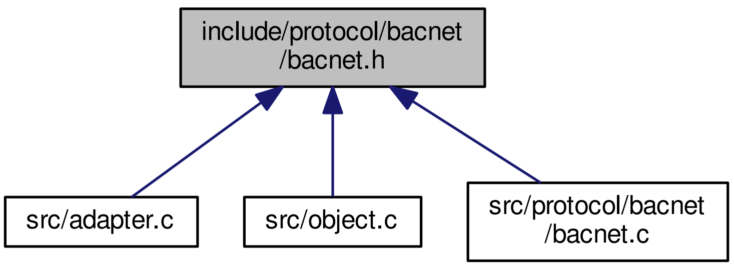
  get_object_queue_sem get the object of work queue semphore

7.13 include/protocol/bacnet/bacnet.h File Reference

```
#include "enum.h"
```
Include dependency graph for bacnet.h:



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



Data Structures

- struct bacnet

bacnet bacnet interface struct

Typedefs

- typedef struct bacnet bacnet_port_handle_t

  bacnet bacnet interface struct

Functions

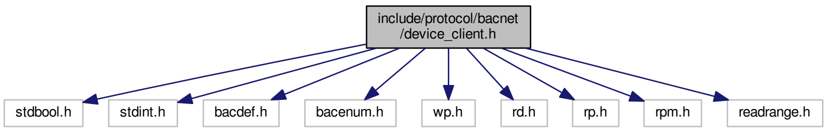
- void ∗ bacnet_work_thread_function (void ∗arg)

  bacnet_work_thread_function The bacnet work thread

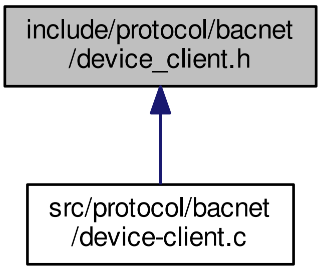
7.14 include/protocol/bacnet/device_client.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "bacdef.h"
#include "bacenum.h"
#include "wp.h"
#include "rd.h"
#include "rp.h"
#include "rpm.h"
#include "readrange.h"
```

Include dependency graph for device_client.h:



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



Data Structures

- struct object_functions
- struct commonBacObj_s
- struct devObj_s

Macros

- #define MAX_DEV_NAME_LEN 32
- #define MAX_DEV_LOC_LEN 64
- #define MAX_DEV_MOD_LEN 32
- #define MAX_DEV_VER_LEN 16
- #define MAX_DEV_DESC_LEN 64

Typedefs

- typedef void(∗ object_init_function)(void)
- typedef unsigned(∗ object_count_function)(void)
- typedef uint32_t(∗ object_index_to_instance_function)(unsigned index)
- typedef bool(∗ object_name_function)(uint32_t object_instance, BACNET_CHARACTER_STRIN←G ∗object_name)
- typedef bool(∗ object_valid_instance_function)(uint32_t object_instance)
- typedef unsigned(∗ object_iterate_function)(unsigned current_index)
- typedef bool(∗ object_value_list_function)(uint32_t object_instance, BACNET_PROPERTY_VALU←E ∗value_list)
- typedef bool(∗ object_cov_function)(uint32_t object_instance)
- typedef void(∗ object_cov_clear_function)(uint32_t object_instance)
- typedef void(∗ object_intrinsic_reporting_function)(uint32_t object_instance)
- typedef struct object_functions object_functions_t
- typedef struct commonBacObj_s COMMON_BAC_OBJECT
- typedef struct devObj_s DEVICE_OBJECT_DATA

Functions

- void Device_Init (object_functions_t ∗object_table)
- bool Device_Reinitialize (BACNET_REINITIALIZE_DEVICE_DATA ∗rd_data)
- BACNET_REINITIALIZED_STATE Device_Reinitialized_State (void)
- rr_info_function Device_Objects_RR_Info (BACNET_OBJECT_TYPE object_type)
- void Device_getCurrentDateTime (BACNET_DATE_TIME ∗DateTime)
- void Device_Property_Lists (const int ∗∗pRequired, const int ∗∗pOptional, const int ∗∗pProprietary)
- void Device_Objects_Property_List (BACNET_OBJECT_TYPE object_type, struct special_property_list_←t ∗pPropertyList)
- bool Device_Encode_Value_List (BACNET_OBJECT_TYPE object_type, uint32_t object_instance, BACN←ET_PROPERTY_VALUE ∗value_list)
- bool Device_Value_List_Supported (BACNET_OBJECT_TYPE object_type)
- bool Device_COV (BACNET_OBJECT_TYPE object_type, uint32_t object_instance)
- void Device_COV_Clear (BACNET_OBJECT_TYPE object_type, uint32_t object_instance)
- uint32_t Device_Object_Instance_Number (void)
- bool Device_Set_Object_Instance_Number (uint32_t object_id)
- bool Device_Valid_Object_Instance_Number (uint32_t object_id)
- unsigned Device_Object_List_Count (void)
- bool Device_Object_List_Identifier (unsigned array_index, int ∗object_type, uint32_t ∗instance)
- unsigned Device_Count (void)

- uint32_t Device_Index_To_Instance (unsigned index)
- bool Device_Object_Name (uint32_t object_instance, BACNET_CHARACTER_STRING ∗object_name)
- bool Device_Set_Object_Name (BACNET_CHARACTER_STRING ∗object_name)
- bool Device_Object_Name_Copy (BACNET_OBJECT_TYPE object_type, uint32_t object_instance, BACN↵ ET_CHARACTER_STRING ∗object_name)
- BACNET_DEVICE_STATUS Device_System_Status (void)
- int Device_Set_System_Status (BACNET_DEVICE_STATUS status, bool local)
- const char ∗ Device_Vendor_Name (void)
- uint16_t Device_Vendor_Identifier (void)
- void Device_Set_Vendor_Identifier (uint16_t vendor_id)
- const char ∗ Device_Model_Name (void)
- bool Device_Set_Model_Name (const char ∗name, size_t length)
- const char ∗ Device_Firmware_Revision (void)
- const char ∗ Device_Application_Software_Version (void)
- bool Device_Set_Application_Software_Version (const char ∗name, size_t length)
- const char ∗ Device_Description (void)
- bool Device_Set_Description (const char ∗name, size_t length)
- const char ∗ Device_Location (void)
- bool Device_Set_Location (const char ∗name, size_t length)
- uint8_t Device_Protocol_Version (void)
- uint8_t Device_Protocol_Revision (void)
- BACNET_SEGMENTATION Device_Segmentation_Supported (void)
- uint32_t Device_Database_Revision (void)
- void Device_Set_Database_Revision (uint32_t revision)
- void Device_Inc_Database_Revision (void)
- bool Device_Valid_Object_Name (BACNET_CHARACTER_STRING ∗object_name, int ∗object_type, uint32_t ∗object_instance)
- bool Device_Valid_Object_Id (int object_type, uint32_t object_instance)
- int Device_Read_Property (BACNET_READ_PROPERTY_DATA ∗rpdata)
- bool Device_Write_Property (BACNET_WRITE_PROPERTY_DATA ∗wp_data)
- bool DeviceGetRRInfo (BACNET_READ_RANGE_DATA ∗pRequest, RR_PROP_INFO ∗pInfo)
- int Device_Read_Property_Local (BACNET_READ_PROPERTY_DATA ∗rpdata)
- bool Device_Write_Property_Local (BACNET_WRITE_PROPERTY_DATA ∗wp_data)
- void Routing_Device_Init (uint32_t first_object_instance)
- uint16_t Add_Routed_Device (uint32_t Object_Instance, BACNET_CHARACTER_STRING ∗Object_Name, const char ∗Description)
- DEVICE_OBJECT_DATA ∗ Get_Routed_Device_Object (int idx)
- BACNET_ADDRESS ∗ Get_Routed_Device_Address (int idx)
- void routed_get_my_address (BACNET_ADDRESS ∗my_address)
- bool Routed_Device_Address_Lookup (int idx, uint8_t address_len, uint8_t ∗mac_adress)
- bool Routed_Device_GetNext (BACNET_ADDRESS ∗dest, int ∗DNET_list, int ∗cursor)
- bool Routed_Device_Is_Valid_Network (uint16_t dest_net, int ∗DNET_list)
- uint32_t Routed_Device_Index_To_Instance (unsigned index)
- bool Routed_Device_Valid_Object_Instance_Number (uint32_t object_id)
- bool Routed_Device_Name (uint32_t object_instance, BACNET_CHARACTER_STRING ∗object_name)
- uint32_t Routed_Device_Object_Instance_Number (void)
- bool Routed_Device_Set_Object_Instance_Number (uint32_t object_id)
- bool Routed_Device_Set_Object_Name (uint8_t encoding, const char ∗value, size_t length)
- bool Routed_Device_Set_Description (const char ∗name, size_t length)
- void Routed_Device_Inc_Database_Revision (void)
- int Routed_Device_Service_Approval (BACNET_CONFIRMED_SERVICE service, int service_argument, uint8_t ∗apdu_buff, uint8_t invoke_id)

7.14.1 Detailed Description

Defines functions for handling all BACnet objects belonging to a BACnet device, as well as Device-specific properties.

Definition in file device_client.h.

7.14.2 Macro Definition Documentation

7.14.2.1 #define MAX_DEV_DESC_LEN 64

Definition at line 174 of file device_client.h.

7.14.2.2 #define MAX_DEV_LOC_LEN 64

Definition at line 171 of file device_client.h.

7.14.2.3 #define MAX_DEV_MOD_LEN 32

Definition at line 172 of file device_client.h.

7.14.2.4 #define MAX_DEV_NAME_LEN 32

Definition at line 170 of file device_client.h.

7.14.2.5 #define MAX_DEV_VER_LEN 16

Definition at line 173 of file device_client.h.

7.14.3 Typedef Documentation

7.14.3.1 typedef struct commonBacObj_s COMMON_BAC_OBJECT

Structure to define the Object Properties common to all Objects.

7.14.3.2 typedef struct devObj_s DEVICE_OBJECT_DATA

Structure to define the Properties of Device Objects which distinguish one instance from another. This structure only defines fields for properties that are unique to a given Device object. The rest may be fixed in device.c or hard-coded into the read-property encoding. This may be useful for implementations which manage multiple Devices, eg, a Gateway.

7.14.3.3 typedef unsigned(∗ object_count_function)(void)

Counts the number of objects of this type.

Returns

Count of implemented objects of this type.

Definition at line 54 of file device_client.h.

7.14.3.4 typedef void($*$ object_cov_clear_function)(uint32_t object_instance)

Look in the table of objects for this instance to clear the changed flag.

Parameters

`in`	*The*	object instance number to be looked up.

Definition at line 130 of file device_client.h.

7.14.3.5 typedef bool(∗ object_cov_function)(uint32_t object_instance)

Look in the table of objects for this instance to see if value changed.

Parameters

`in`	*The*	object instance number to be looked up.

Returns

True if the object instance has changed.

Definition at line 122 of file device_client.h.

7.14.3.6 typedef struct object_functions object_functions_t

Defines the group of object helper functions for any supported Object.

Each Object must provide some implementation of each of these helpers in order to properly support the handlers. Eg, the ReadProperty handler handler_read_property() relies on the instance of Object_Read_Property for each Object type, or configure the function as NULL. In both appearance and operation, this group of functions acts like they are member functions of a C++ Object base class.

7.14.3.7 typedef uint32_t(∗ object_index_to_instance_function)(unsigned index)

Maps an object index position to its corresponding BACnet object instance number.

Parameters

index	[in] The index of the object, in the array of objects of its type.

Returns

The BACnet object instance number to be used in a BACNET_OBJECT_ID.

Definition at line 64 of file device_client.h.

7.14.3.8 typedef void(∗ object_init_function)(void)

Called so a BACnet object can perform any necessary initialization.

Definition at line 46 of file device_client.h.

7.14.3.9 typedef void(∗ object_intrinsic_reporting_function)(uint32_t object_instance)

Intrinsic Reporting funcionality.

Parameters

`in`	*Object*	instance.

Definition at line 138 of file device_client.h.

7.14.3.10 typedef unsigned(∗ object_iterate_function)(unsigned current_index)

Helper function to step through an array of objects and find either the first one or the next one of a given type. Used to step through an array of objects which is not necessarily contiguious for each type i.e. the index for the 'n'th object of a given type is not necessarily 'n'.

Parameters

`in`	*The*	index of the current object or a value of ∼0 to indicate start at the beginning.

Returns

The index of the next object of the required type or ∼0 (all bits == 1) to indicate no more objects found.

Definition at line 102 of file device_client.h.

7.14.3.11 typedef bool(∗ object_name_function)(uint32_t object_instance, BACNET_CHARACTER_STRING ∗object_name)

Provides the BACnet Object_Name for a given object instance of this type.

Parameters

object_instance	[in] The object instance number to be looked up.
object_name	[in,out] Pointer to a character_string structure that will hold a copy of the object name if this is a valid object_instance.

Returns

True if the object_instance is valid and object_name has been filled with a copy of the Object's name.

Definition at line 77 of file device_client.h.

7.14.3.12 typedef bool(∗ object_valid_instance_function)(uint32_t object_instance)

Look in the table of objects of this type, and see if this is a valid instance number.

Parameters

`in`	*The*	object instance number to be looked up.

Returns

True if the object instance refers to a valid object of this type.

Definition at line 88 of file device_client.h.

7.14.3.13 typedef bool(∗ object_value_list_function)(uint32_t object_instance, BACNET_PROPERTY_VALUE ∗value_list)

Look in the table of objects of this type, and get the COV Value List.

Parameters

in	*The*	object instance number to be looked up.
out	*The*	value list

Returns

True if the object instance supports this feature, and has changed.

Definition at line 112 of file device_client.h.

7.14.4 Function Documentation

7.14.4.1 uint16_t Add_Routed_Device (uint32_t *Object_Instance,* BACNET_CHARACTER_STRING ∗ *Object_Name,* const char ∗ *Description*)

7.14.4.2 const char∗ Device_Application_Software_Version (void)

Definition at line 362 of file device-client.c.

7.14.4.3 unsigned Device_Count (void)

Definition at line 163 of file device-client.c.

7.14.4.4 bool Device_COV (BACNET_OBJECT_TYPE *object_type,* uint32_t *object_instance*)

7.14.4.5 void Device_COV_Clear (BACNET_OBJECT_TYPE *object_type,* uint32_t *object_instance*)

7.14.4.6 uint32_t Device_Database_Revision (void)

Definition at line 443 of file device-client.c.

7.14.4.7 const char∗ Device_Description (void)

Definition at line 383 of file device-client.c.

7.14.4.8 bool Device_Encode_Value_List (BACNET_OBJECT_TYPE *object_type,* uint32_t *object_instance,* BACNET_PROPERTY_VALUE ∗ *value_list*)

7.14.4.9 const char∗ Device_Firmware_Revision (void)

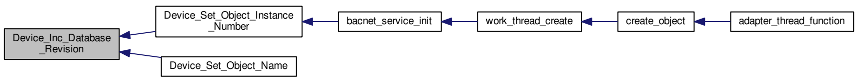
Definition at line 356 of file device-client.c.

7.14.4.10 void Device_getCurrentDateTime (BACNET_DATE_TIME ∗ *DateTime*)

7.14.4.11 void Device_Inc_Database_Revision (void)

Definition at line 460 of file device-client.c.

Here is the caller graph for this function:



7.14.4.12 uint32_t Device_Index_To_Instance (unsigned *index*)

Definition at line 169 of file device-client.c.

7.14.4.13 void Device_Init (object_functions_t ∗ *object_table*)

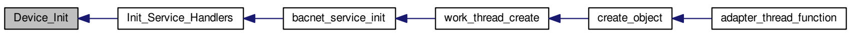
Initialize the Device Object. Initialize the group of object helper functions for any supported Object. Initialize each of the Device Object child Object instances.

Parameters

object_table	[in,out] array of structure with object functions. Each Child Object must provide some implementation of each of these functions in order to properly support the default handlers.

Definition at line 895 of file device-client.c.

Here is the caller graph for this function:



7.14.4.14 const char∗ Device_Location (void)

Definition at line 404 of file device-client.c.

7.14.4.15 const char∗ Device_Model_Name (void)

Definition at line 335 of file device-client.c.

7.14.4.16 uint32_t Device_Object_Instance_Number (void)

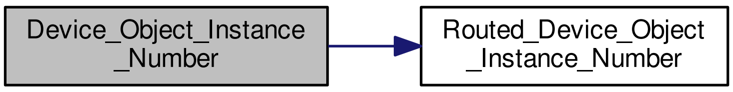
Return the Object Instance number for our (single) Device Object. This is a key function, widely invoked by the handler code, since it provides "our" (ie, local) address.

Returns

The Instance number used in the BACNET_OBJECT_ID for the Device.

Definition at line 184 of file device-client.c.

Here is the call graph for this function:


Device_Object_Instance_Number
Routed_Device_Object_Instance_Number

7.14.4.17 unsigned Device_Object_List_Count (void)

Get the total count of objects supported by this Device Object.

Note

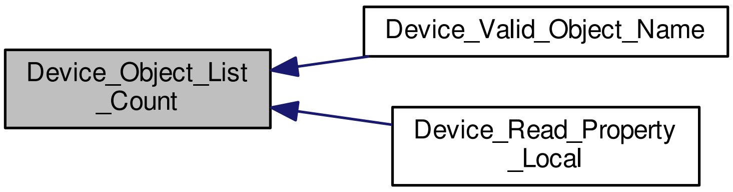
Since many network clients depend on the object list for discovery, it must be consistent!

Returns

The count of objects, for all supported Object types.

Definition at line 471 of file device-client.c.

Here is the caller graph for this function:



7.14.4.18 bool Device_Object_List_Identifier (unsigned *array_index,* int ∗ *object_type,* uint32_t ∗ *instance*)

Lookup the Object at the given array index in the Device's Object List. Even though we don't keep a single linear array of objects in the Device, this method acts as though we do and works through a virtual, concatenated array of all of our object type arrays.

Parameters

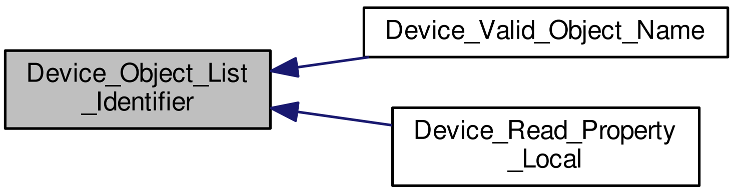
array_index	[in] The desired array index (1 to N)
object_type	[out] The object's type, if found.
instance	[out] The object's instance number, if found.

Returns

True if found, else false.

Definition at line 499 of file device-client.c.

Here is the caller graph for this function:



7.14.4.19 bool Device_Object_Name (uint32_t *object_instance,* BACNET_CHARACTER_STRING ∗ *object_name*)

Definition at line 215 of file device-client.c.

7.14.4.20 bool Device_Object_Name_Copy (BACNET_OBJECT_TYPE *object_type,* uint32_t *object_instance,* BACNET_CHARACTER_STRING ∗ *object_name*)

Copy a child object's object_name value, given its ID.

Parameters

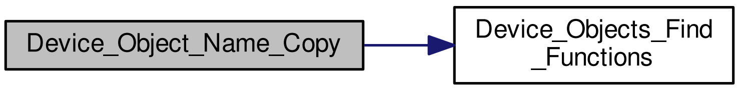
object_type	[in] The BACNET_OBJECT_TYPE of the child Object.
object_instance	[in] The object instance number of the child Object.
object_name	[out] The Object Name found for this child Object.

Returns

True on success or else False if not found.

Definition at line 620 of file device-client.c.

Here is the call graph for this function:



7.14.4.21 void Device_Objects_Property_List (BACNET_OBJECT_TYPE *object_type,* struct special_property_list_t ∗ *pPropertyList*)

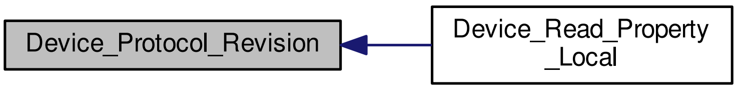
7.14.4.22 rr_info_function Device_Objects_RR_Info (BACNET_OBJECT_TYPE *object_type*)

7.14.4.23 void Device_Property_Lists (const int ∗∗ *pRequired,* const int ∗∗ *pOptional,* const int ∗∗ *pProprietary*)

7.14.4.24 uint8_t Device_Protocol_Revision (void)

Definition at line 431 of file device-client.c.

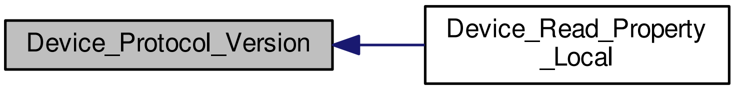
Here is the caller graph for this function:



7.14.4.25 uint8_t Device_Protocol_Version (void)

Definition at line 425 of file device-client.c.

Here is the caller graph for this function:



7.14.4.26 int Device_Read_Property (BACNET_READ_PROPERTY_DATA ∗ *rpdata*)

Looks up the requested Object and Property, and encodes its Value in an APDU.

If the Object or Property can't be found, sets the error class and code.

Parameters

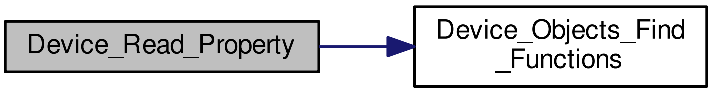
rpdata	[in,out] Structure with the desired Object and Property info on entry, and APDU message on return.

Returns

The length of the APDU on success, else BACNET_STATUS_ERROR

Definition at line 859 of file device-client.c.

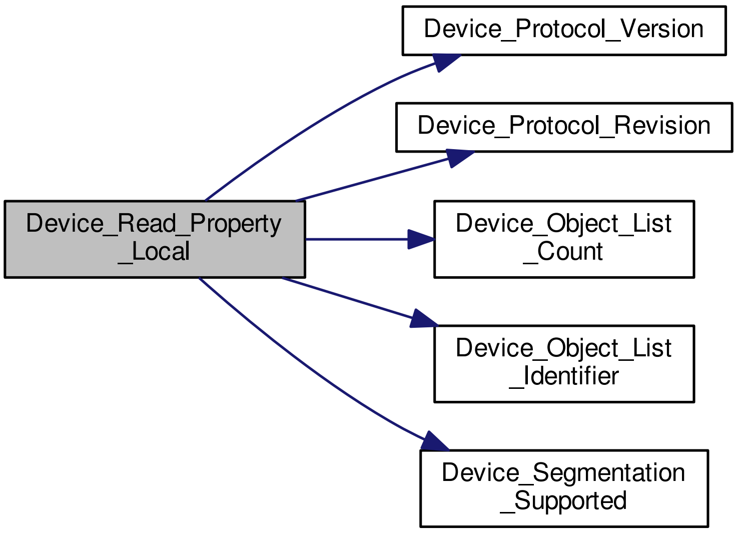
Here is the call graph for this function:



7.14.4.27 int Device_Read_Property_Local (BACNET_READ_PROPERTY_DATA ∗ *rpdata*)

Definition at line 638 of file device-client.c.

Here is the call graph for this function:


Device_Read_Property _Local
Device_Protocol_Version
Device_Protocol_Revision
Device_Object_List _Count
Device_Object_List _Identifier
Device_Segmentation _Supported

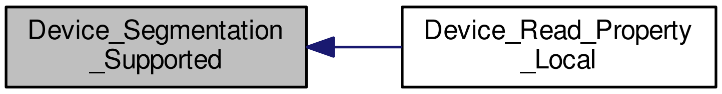
7.14.4.28 bool Device_Reinitialize (BACNET_REINITIALIZE_DEVICE_DATA ∗ *rd_data*)

7.14.4.29 BACNET_REINITIALIZED_STATE Device_Reinitialized_State (void)

7.14.4.30 BACNET_SEGMENTATION Device_Segmentation_Supported (void)

Definition at line 437 of file device-client.c.

Here is the caller graph for this function:



7.14.4.31 bool Device_Set_Application_Software_Version (const char ∗ *name,* size_t *length*)

Definition at line 368 of file device-client.c.

7.14.4.32 void Device_Set_Database_Revision (uint32_t *revision*)

Definition at line 449 of file device-client.c.

7.14.4.33 bool Device_Set_Description (const char ∗ *name,* size_t *length*)

Definition at line 389 of file device-client.c.

7.14.4.34 bool Device_Set_Location (const char ∗ *name,* size_t *length*)

Definition at line 410 of file device-client.c.

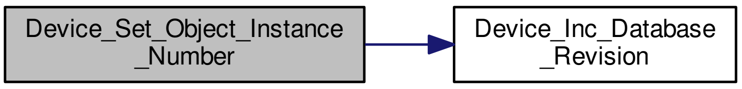
7.14.4.35 bool Device_Set_Model_Name (const char ∗ *name,* size_t *length*)

Definition at line 341 of file device-client.c.

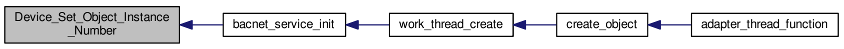
7.14.4.36 bool Device_Set_Object_Instance_Number (uint32_t *object_id*)

Definition at line 194 of file device-client.c.

Here is the call graph for this function:



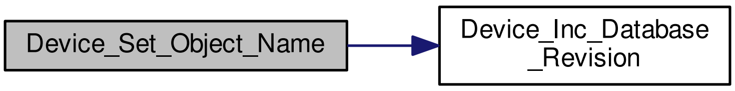
Here is the caller graph for this function:



7.14.4.37 bool Device_Set_Object_Name (BACNET_CHARACTER_STRING ∗ *object_name*)

Definition at line 228 of file device-client.c.

Here is the call graph for this function:



7.14.4.38 int Device_Set_System_Status (BACNET_DEVICE_STATUS *status,* bool *local*)

Definition at line 248 of file device-client.c.

7.14.4.39 void Device_Set_Vendor_Identifier (uint16_t *vendor_id*)

Definition at line 329 of file device-client.c.

7.14.4.40 BACNET_DEVICE_STATUS Device_System_Status (void)

Definition at line 242 of file device-client.c.

7.14.4.41 bool Device_Valid_Object_Id (int *object_type,* uint32_t *object_instance*)

Determine if we have an object of this type and instance number.

Parameters

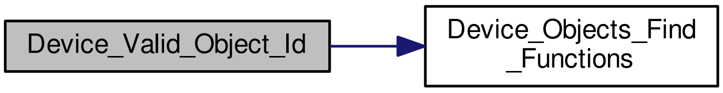
object_type	[in] The desired BACNET_OBJECT_TYPE
object_instance	[in] The object instance number to be looked up.

Returns

True if found, else False if no such Object in this device.

Definition at line 599 of file device-client.c.

Here is the call graph for this function:



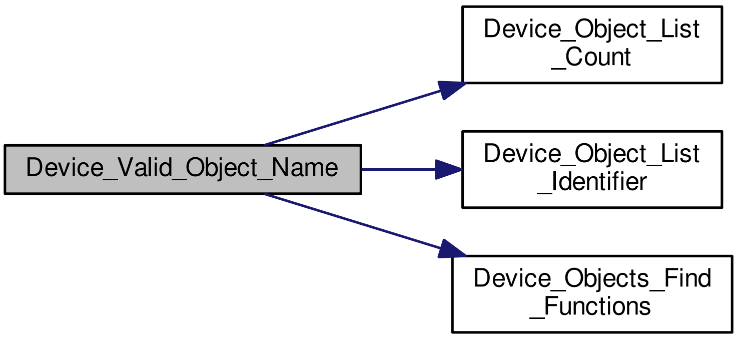
7.14.4.42 bool Device_Valid_Object_Instance_Number (uint32_t *object_id*)

Definition at line 209 of file device-client.c.

7.14.4.43 bool Device_Valid_Object_Name (BACNET_CHARACTER_STRING ∗ *object_name,* int ∗ *object_type,* uint32_t ∗ *object_instance*)

Definition at line 558 of file device-client.c.

Here is the call graph for this function:



7.14.4.44 bool Device_Value_List_Supported (BACNET_OBJECT_TYPE *object_type*)

7.14.4.45 uint16_t Device_Vendor_Identifier (void)

Returns the Vendor ID for this Device. See the assignments at `http://www.bacnet.org/VendorID/BACnet%20Vendor%20IDs.htm`

Returns

The Vendor ID of this Device.

Definition at line 323 of file device-client.c.

7.14.4.46 const char∗ Device_Vendor_Name (void)

Definition at line 313 of file device-client.c.

7.14.4.47 bool Device_Write_Property (BACNET_WRITE_PROPERTY_DATA ∗ *wp_data*)

7.14.4.48 bool Device_Write_Property_Local (BACNET_WRITE_PROPERTY_DATA ∗ *wp_data*)

7.14.4.49 bool DeviceGetRRInfo (BACNET_READ_RANGE_DATA ∗ *pRequest,* RR_PROP_INFO ∗ *pInfo*)

7.14.4.50 BACNET_ADDRESS∗ Get_Routed_Device_Address (int *idx*)

7.14.4.51 DEVICE_OBJECT_DATA∗ Get_Routed_Device_Object (int *idx*)

7.14.4.52 bool Routed_Device_Address_Lookup (int *idx,* uint8_t *address_len,* uint8_t ∗ *mac_adress*)

7.14.4.53 bool Routed_Device_GetNext (BACNET_ADDRESS ∗ *dest,* int ∗ *DNET_list,* int ∗ *cursor*)

7.14.4.54 void Routed_Device_Inc_Database_Revision (void)

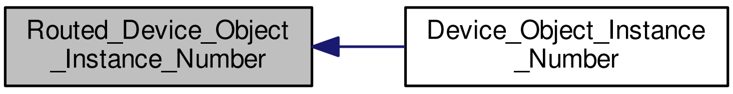
7.14.4.55 uint32_t Routed_Device_Index_To_Instance (unsigned *index*)

7.14.4.56 bool Routed_Device_Is_Valid_Network (uint16_t *dest_net,* int ∗ *DNET_list*)

7.14.4.57 bool Routed_Device_Name (uint32_t *object_instance,* BACNET_CHARACTER_STRING ∗ *object_name*)

7.14.4.58 uint32_t Routed_Device_Object_Instance_Number (void)

Here is the caller graph for this function:


Routed_Device_Object _Instance_Number
Device_Object_Instance _Number

7.14.4.59 int Routed_Device_Service_Approval (BACNET_CONFIRMED_SERVICE *service,* int *service_argument,* uint8_t ∗ *apdu_buff,* uint8_t *invoke_id*)

7.14.4.60 bool Routed_Device_Set_Description (const char ∗ *name,* size_t *length*)

7.14.4.61 bool Routed_Device_Set_Object_Instance_Number (uint32_t *object_id*)

7.14.4.62 bool Routed_Device_Set_Object_Name (uint8_t *encoding,* const char ∗ *value,* size_t *length*)

7.14.4.63 bool Routed_Device_Valid_Object_Instance_Number (uint32_t *object_id*)

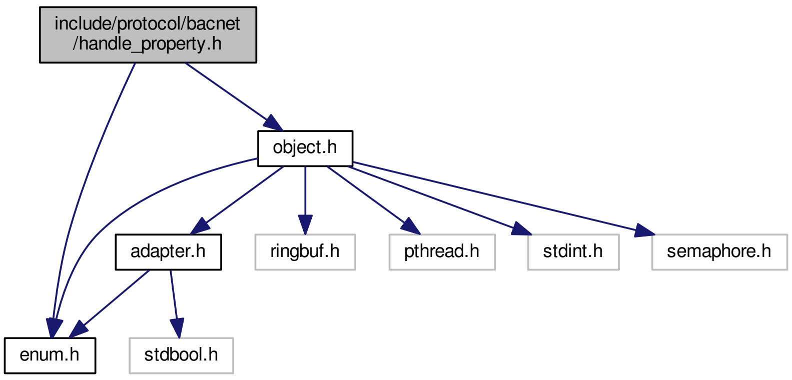
7.14.4.64 void routed_get_my_address (BACNET_ADDRESS ∗ *my_address*)

7.14.4.65 void Routing_Device_Init (uint32_t *first_object_instance*)

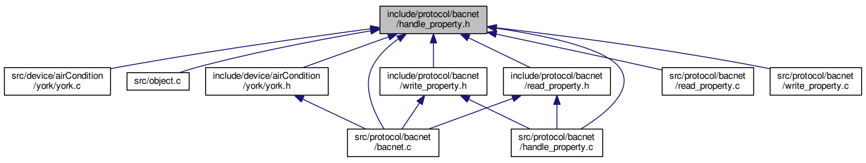
7.15 include/protocol/bacnet/handle_property.h File Reference

```
#include "enum.h"
#include "object.h"
```

Include dependency graph for handle_property.h:



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



Data Structures

- struct bacnet_write_args_t

  bacnet write arg struct
- struct bacnet_read_args_t

  bacnet read property struct

Macros

- #define BACNET_READ_ARGS_OBJECT_MAX 10

Functions

- int get_air_condition_bacnet_write_args (bacnet_write_args_t ∗args, unsigned int device_id, int command)

  get_air_condition_bacnet_write_args bacnet write args
- int get_air_condition_bacnet_read_args (bacnet_read_args_t ∗args, unsigned int device_id)

get_air_condition_bacnet_read_args bacnet read args

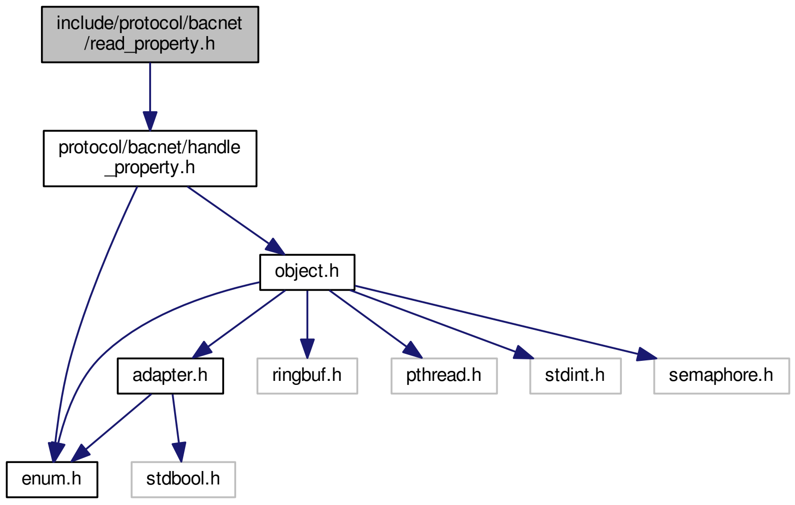
- int bacnet_service_init (object_management_t ∗adapter)

  bacnet_service_init bacnet physics initialze.

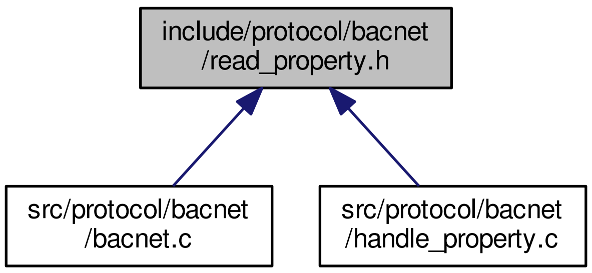
7.16 include/protocol/bacnet/read_property.h File Reference

```
#include "protocol/bacnet/handle_property.h"
```

Include dependency graph for read_property.h:



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



Functions

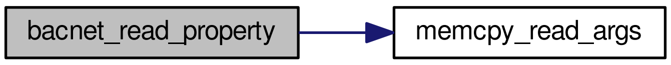
- int bacnet_read_property (bacnet_read_args_t ∗args)

7.16.1 Function Documentation

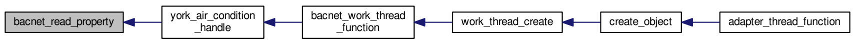
7.16.1.1 int bacnet_read_property (**bacnet_read_args_t** ∗ *args*)

Definition at line 152 of file read_property.c.

Here is the call graph for this function:



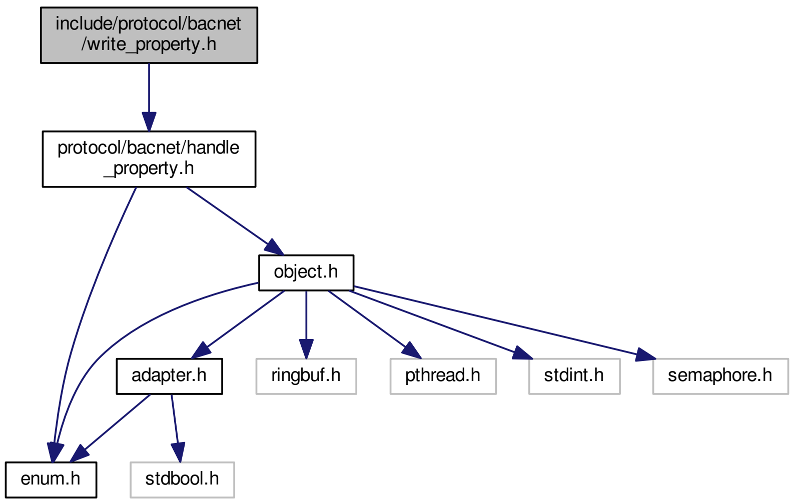
Here is the caller graph for this function:



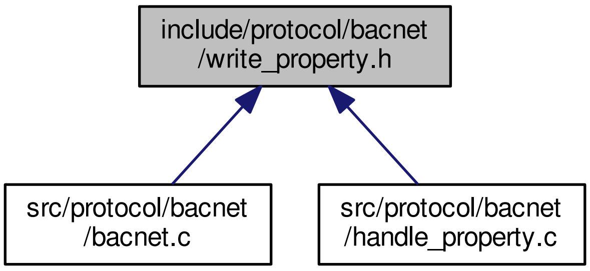
7.17 include/protocol/bacnet/write_property.h File Reference

```
#include "protocol/bacnet/handle_property.h"
```

Include dependency graph for write_property.h:


include/protocol/bacnet/write_property.h
protocol/bacnet/handle_property.h
object.h
adapter.h
ringbuf.h
pthread.h
stdint.h
semaphore.h
enum.h
stdbool.h

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



Functions

- int bacnet_write_property (const bacnet_write_args_t ∗args)

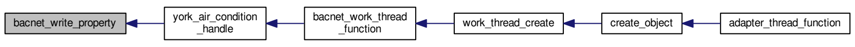
7.17.1 Function Documentation

7.17.1.1 int bacnet_write_property (const **bacnet_write_args_t** ∗ *args*)

500ms∗4, 2s

Definition at line 63 of file write_property.c.

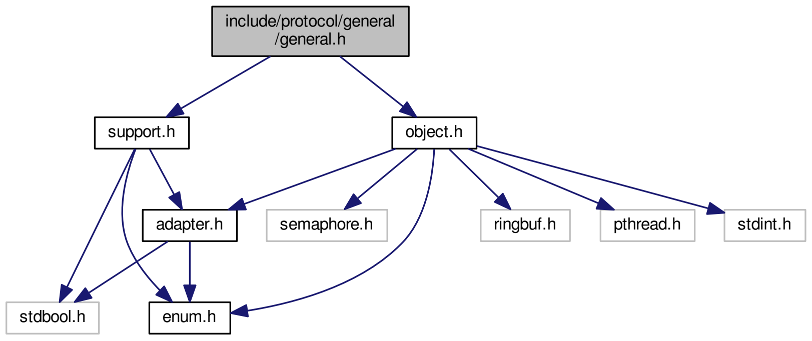
Here is the caller graph for this function:



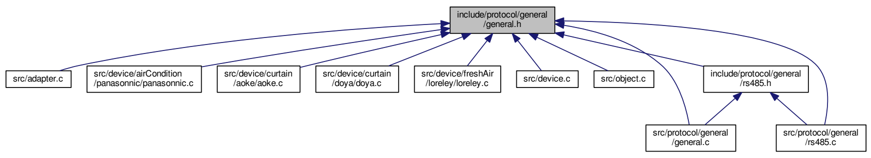
7.18 include/protocol/general/general.h File Reference

```
#include "support.h"
#include "object.h"
```

Include dependency graph for general.h:



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



Data Structures

- struct mstp_port_handle

  mstp_port_handle general protocol(user defined)

Typedefs

- typedef struct mstp_port_handle mstp_port_handle_t

  mstp_port_handle general protocol(user defined)

Functions

- int general_service_init (object_management_t ∗object)

  general_service_init The general protocol(user defined) initilize

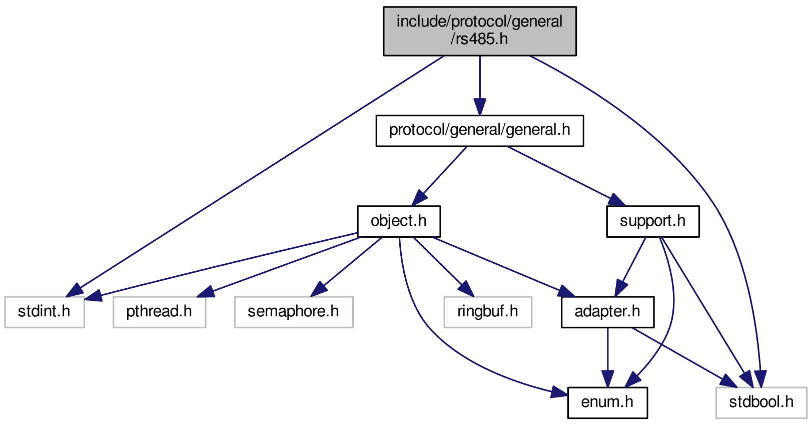
- void ∗ general_work_thread_function (void ∗arg)

  general_work_thread_function The general work thread function

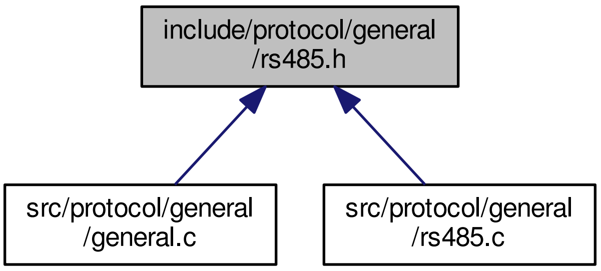
7.19 include/protocol/general/rs485.h File Reference

```
#include <stdint.h>
#include <stdbool.h>
#include "protocol/general/general.h"
```

Include dependency graph for rs485.h:



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



Functions

- void rs485_set_interface (char ∗ifname)

  RS485_Set_Interface rs485 interface name.
- const char ∗ rs485_get_interface (void)

  RS485_Get_Interface get the rs485 interface name.
- void rs485_initialize (void)

  RS485_Initialize.
- int rs485_send_handle_frame (volatile struct mstp_port_handle ∗mstp_port)

  rs485_send_handle_frame rs485 bus package a send frame, and send the package to bus.
- int rs485_recv_handle_frame (volatile struct mstp_port_handle ∗mstp_port)

  rs485_recv_handle_frame rs485 bus receive a frame, and call process these data.
- bool rs485_set_baud_rate (uint32_t baud)

  RS485_Set_Baud_Rate set the rs485 buad rate.
- void rs485_cleanup (void)

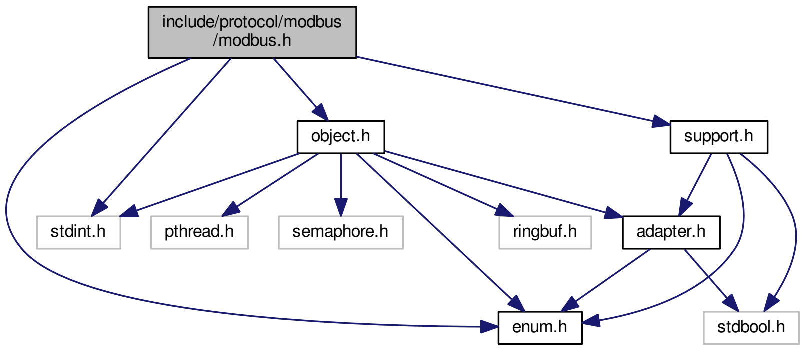
  RS485_Cleanup The rs485 initaialize fail, have clean.

7.20 include/rs485.h File Reference

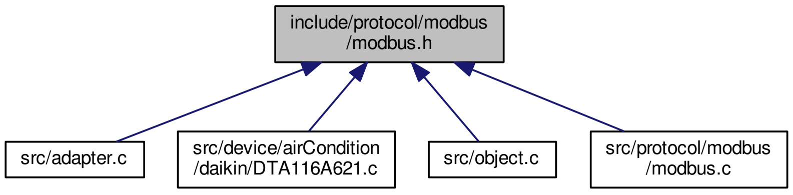
7.21 include/protocol/modbus/modbus.h File Reference

```
#include <stdint.h>
#include "enum.h"
#include "object.h"
#include "support.h"
```

Include dependency graph for modbus.h:



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


include/protocol/modbus /modbus.h
src/adapter.c
src/device/airCondition /daikin/DTA116A621.c
src/object.c
src/protocol/modbus /modbus.c

Data Structures

- struct modbus_port_handle_t

  The modbus port interface.

Functions

- void ∗ modbus_work_thread_function (void ∗arg)

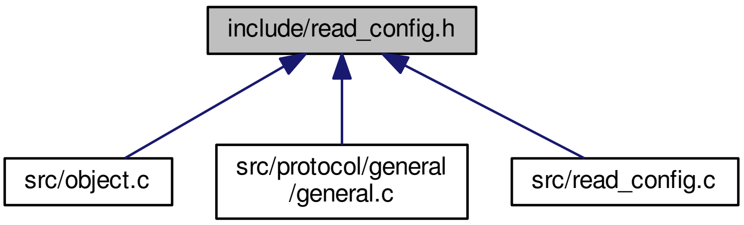
  modbus_work_thread_function The modbus work thread
- int modbus_service_init (object_management_t ∗object)

  modbus_service_init The modbus interface intialize.
- void modbus_service_deinit (object_management_t ∗object)

  modbus_service_deinit clean the modbus service, The haved called by thread have exit.

7.22 include/read_config.h File Reference

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



Variables

- int glb_config_general_work_queue_depth
- int glb_config_bacnet_work_queue_depth
- int glb_config_modbus_work_queue_depth
- int glb_config_adapter_message_queue_depth
- int glb_config_general_work_package_mtu

7.22.1 Variable Documentation

7.22.1.1 int glb_config_adapter_message_queue_depth

Definition at line 32 of file read_config.c.

7.22.1.2 int glb_config_bacnet_work_queue_depth

Definition at line 26 of file read_config.c.

7.22.1.3 int glb_config_general_work_package_mtu

Definition at line 23 of file read_config.c.

7.22.1.4 int glb_config_general_work_queue_depth

Definition at line 21 of file read_config.c.

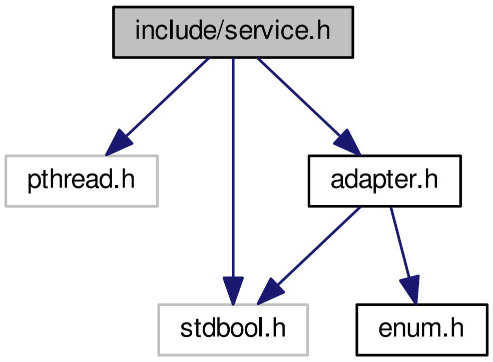
7.22.1.5 int glb_config_modbus_work_queue_depth

Definition at line 29 of file read_config.c.

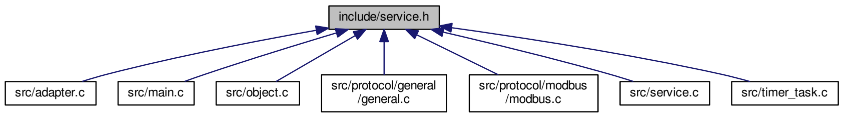
7.23 include/service.h File Reference

```
#include <pthread.h>
#include <stdbool.h>
#include "adapter.h"
```

Include dependency graph for service.h:



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



Data Structures

- struct thread_pool_t

  define the thread pool struct

Functions

- int rs485_service_start (void)

  rs485_service_start The rs485 service start
- int rs485_send_msg_to_client (int clifd, void ∗buffer, int buffer_len)

  rs485_send_msg_to_client send The message to a client
- int rs485_recv_msg_from_client (int clifd, void ∗buffer, int buffer_len)

  rs485_recv_msg_from_client recvieve a message from client
- int send_msg_to_adapter (const adapter_t ∗adapter)

  send_msg_to_adapter send a message to self,

7.23.1 Detailed Description

www.enno.com

Date

: Mar 15, 2016

Author

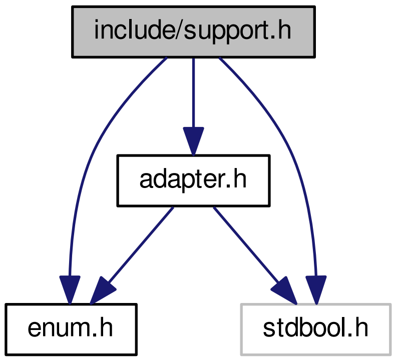
: wong

Definition in file service.h.

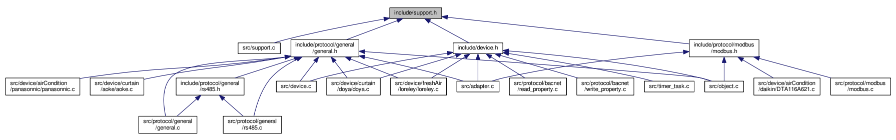
7.24 include/support.h File Reference

```
#include "enum.h"
#include "adapter.h"
#include <stdbool.h>
```

Include dependency graph for support.h:



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


include/support.h
src/support.c
include/protocol/general/general.h
include/device.h
include/protocol/modbus/modbus.h
src/device/airCondition/panasonnic/panasonnic.c
src/device/curtain/aoke/aoke.c
include/protocol/general/rs485.h
src/device.c
src/device/curtain/doya/doya.c
src/device/freshAir/loreley/loreley.c
src/adapter.c
src/protocol/bacnet/read_property.c
src/protocol/bacnet/write_property.c
src/timer_task.c
src/object.c
src/device/airCondition/daikin/DTA116A621.c
src/protocol/modbus/modbus.c
src/protocol/general/general.c
src/protocol/general/rs485.c

Data Structures

- struct device_profile

  device_profile device process method

Typedefs

- typedef int(∗ method_send)(volatile void ∗context)

  int you have full the context pointer.
- typedef int(∗ method_recv)(volatile void ∗context)

  int you have full the context pointer.

Functions

- bool check_device_support (const adapter_t ∗adatper)

  check_device_support check the device have supported by rs485 service
- struct device_profile ∗ get_support_device_profile (rs485_factory_name_enum name)

  get_support_device_profile Get the device profile, The struct device_profile
- int get_support_device_profile_numbers (rs485_factory_name_enum name)

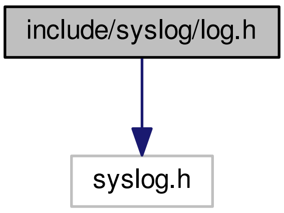
  get_support_device_profile_numbers Get the device profile have support how many command.
- method_send get_device_send_package_function (const struct device_profile ∗profile, int profile_numbers, int command)

  get_device_send_package_function Get the device profile send package callback function
- method_recv get_device_recv_package_function (const struct device_profile ∗profile, int profile_numbers, int command)

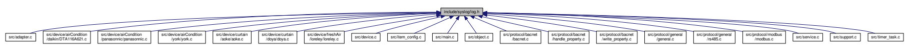
  get_device_recv_package_function Get the device profile receive package callback function

7.25 include/syslog/log.h File Reference

```
#include <syslog.h>
```
Include dependency graph for log.h:



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


include/syslog/log.h
src/adapter.c
src/device/airCondition/daikin/DTA116A621.c
src/device/airCondition/panasonnic/panasonnic.c
src/device/airCondition/york/york.c
src/device/curtain/aoke/aoke.c
src/device/curtain/doya/doya.c
src/device/freshAir/loreley/loreley.c
src/device.c
src/item_config.c
src/main.c
src/object.c
src/protocol/bacnet/bacnet.c
src/protocol/bacnet/handle_property.c
src/protocol/bacnet/write_property.c
src/protocol/general/general.c
src/protocol/general/rs485.c
src/protocol/modbus/modbus.c
src/service.c
src/support.c
src/timer_task.c

Macros

- #define DEBUG_LINUX_SYSLOG (0)
- #define DEBUG_PRINTF (1)

- #define syslog_error(_fmt_,...) printf(_fmt_",file:%s,line:%d\n", ##__VA_ARGS__, __FILE__, __LINE__)
- #define syslog_warning(_fmt_,...) printf(_fmt_",file:%s,line:%d\n", ##__VA_ARGS__, __FILE__, __LINE__)
- #define syslog_info(_fmt_,...) printf(_fmt_"\n", ##__VA_ARGS__)
- #define syslog_debug(_fmt_,...) printf(_fmt_"\n", ##__VA_ARGS__)
- #define syslog_format printf

7.25.1 Detailed Description

www.enno.com

Date

: Mar 15, 2016

Author

: wong

Definition in file log.h.

7.25.2 Macro Definition Documentation

7.25.2.1 #define DEBUG_LINUX_SYSLOG (0)

Definition at line 18 of file log.h.

7.25.2.2 #define DEBUG_PRINTF (1)

Definition at line 19 of file log.h.

7.25.2.3 #define syslog_debug(*_fmt_,* ...) printf(_fmt_"\n", ##__VA_ARGS__)

Definition at line 38 of file log.h.

7.25.2.4 #define syslog_error(*_fmt_,* ...) printf(_fmt_",file:%s,line:%d\n", ##__VA_ARGS__, __FILE__, __LINE__)

Definition at line 35 of file log.h.

7.25.2.5 #define syslog_format printf

Definition at line 39 of file log.h.

7.25.2.6 #define syslog_info(*_fmt_,* ...) printf(_fmt_"\n", ##__VA_ARGS__)

Definition at line 37 of file log.h.

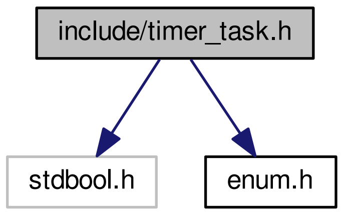
7.25.2.7 #define syslog_warning(*_fmt_,* ...) printf(_fmt_",file:%s,line:%d\n", ##__VA_ARGS__, __FILE__, __LINE__)

Definition at line 36 of file log.h.

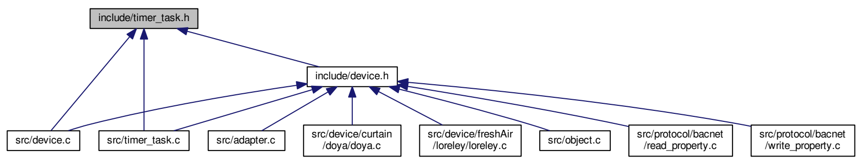
7.26 include/timer_task.h File Reference

```
#include <stdbool.h>
#include "enum.h"
```

Include dependency graph for timer_task.h:



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


include/timer_task.h
include/device.h
src/device.c
src/timer_task.c
src/adapter.c
src/device/curtain /doya/doya.c
src/device/freshAir /loreley/loreley.c
src/object.c
src/protocol/bacnet /read_property.c
src/protocol/bacnet /write_property.c

Data Structures

- struct timer_task_t

    timer task struct

Typedefs

- typedef int(∗ timer_proc_func)(int device_id, int command)

Functions

- void ∗ timer_task_thread_function (void ∗arg)

    timer_task_thread_function The timer task therad start function, just return when the have a error
- int create_device_timer_task (timer_task_t ∗task)

    create_deivce_timer_task create a device timer task , The timer task min tick is 10 second
- int delete_device_timer_task (timer_task_t ∗task)

    delete_device_timer_task delete a device timer task from The timer list.
- int device_timer_task_handle_demo (int device_id, int command)

    device_timer_task_handle_demo timer task handle fucntion demo
- int device_timer_task_handle_curtain_init (int device_id, int command)
- int device_timer_task_handle_curtain_aoke_init (int device_id, int command)
- int device_timer_task_handle_curtain_doya_init (int device_id, int command)

7.26.1 Detailed Description

www.enno.com

Date

: Mar 15, 2016

Author

: wong

Definition in file timer_task.h.

7.26.2 Function Documentation

7.26.2.1 int device_timer_task_handle_curtain_aoke_init (int *device_id,* int *command*)

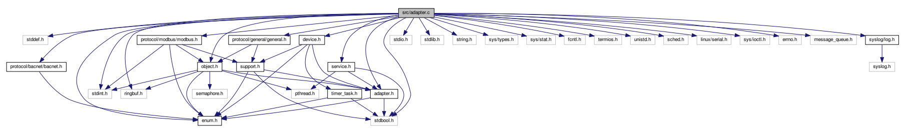
7.26.2.2 int device_timer_task_handle_curtain_doya_init (int *device_id,* int *command*)

7.26.2.3 int device_timer_task_handle_curtain_init (int *device_id,* int *command*)

7.27 src/adapter.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <sched.h>
#include <linux/serial.h>
#include <sys/ioctl.h>
#include <errno.h>
#include "enum.h"
#include "adapter.h"
#include "message_queue.h"
#include "syslog/log.h"
#include "device.h"
#include "object.h"
#include "service.h"
#include "protocol/bacnet/bacnet.h"
#include "protocol/modbus/modbus.h"
#include "protocol/general/general.h"
#include "ringbuf.h"
```

Include dependency graph for adapter.c:



Macros

- #define ADAPTER_MESSAGE_QUEUE_MAX_DEPTH (48)

Functions

- static int adapter_thread_init (void)

  adapter_thread_init initialze the adapter thread , and mesesage queue initial.
- static int process_write_value_service (const adapter_t ∗adapter)

  process_write_value_service process the client write value to device service
- static int process_read_value_service (adapter_t ∗adapter)

  process_read_value_service process the client read value from device service
- void ∗ adapter_thread_function (void ∗arg)

Variables

- static adapter_thread_status_enum adapter_thread_status = ADAPTER_THREAD_STATUS_START
- static adapter_t reply_client
- static bacnet_port_handle_t bacnet
- static modbus_port_handle_t modbus
- static mstp_port_handle_t general

7.27.1 Detailed Description

www.enno.com

Date

: Mar 14, 2016

Author

: chuanjiang.wong

Definition in file adapter.c.

7.27.2 Macro Definition Documentation

7.27.2.1 #define ADAPTER_MESSAGE_QUEUE_MAX_DEPTH (48)

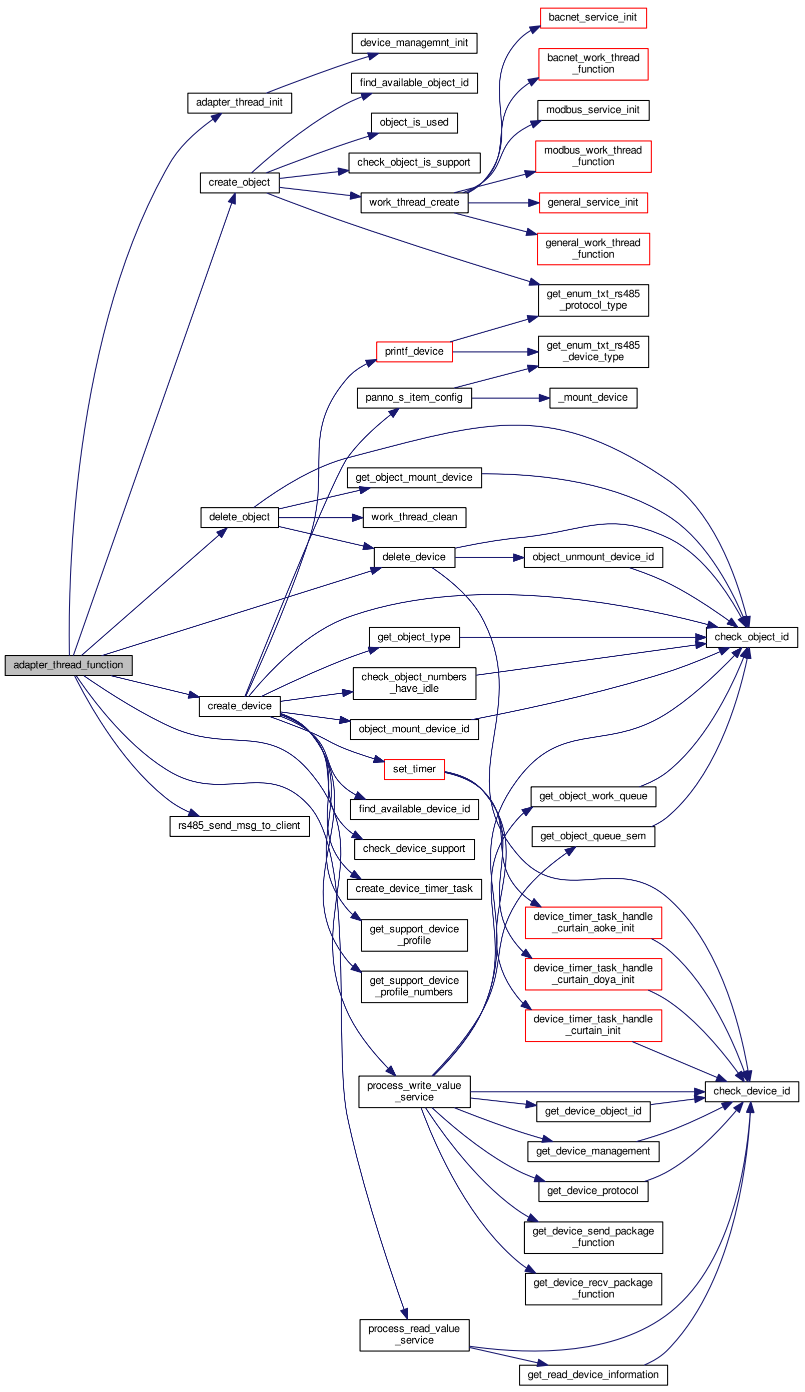
Definition at line 46 of file adapter.c.

7.27.3 Function Documentation

7.27.3.1 void∗ adapter_thread_function (void ∗ *arg*)

Definition at line 300 of file adapter.c.

Here is the call graph for this function:



7.27.4 Variable Documentation

7.27.4.1 adapter_thread_status_enum adapter_thread_status = ADAPTER_THREAD_STATUS_START [static]

define the thread status

Definition at line 51 of file adapter.c.

7.27.4.2 bacnet_port_handle_t bacnet [static]

The bacnet interface

Definition at line 76 of file adapter.c.

7.27.4.3 mstp_port_handle_t general [static]

The general interface

Definition at line 80 of file adapter.c.

7.27.4.4 modbus_port_handle_t modbus [static]

The modbus interface

Definition at line 78 of file adapter.c.

7.27.4.5 adapter_t reply_client [static]

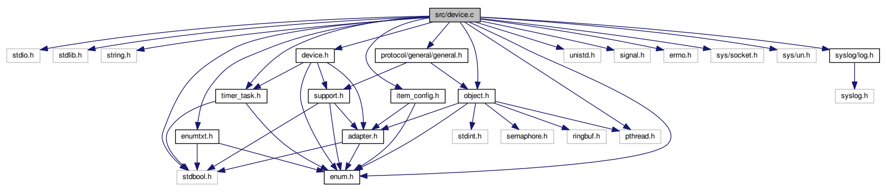
define the message queue

Definition at line 53 of file adapter.c.

7.28 src/device.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <pthread.h>
#include "device.h"
#include "enum.h"
#include "syslog/log.h"
#include "timer_task.h"
#include "object.h"
#include "protocol/general/general.h"
#include "enumtxt.h"
#include "item_config.h"
```

Include dependency graph for device.c:



Macros

- #define RS485_DEVICE_MAX_NUMBERS (100)
- #define RS485_DEVICE_NAME_MAX_LENGTH (108)
- #define RS485_CURTAIN_MAX_FACTORY (16)

Functions

- static bool find_curtain_factory_is_save (rs485_factory_name_enum factory)
- static int find_available_device_id (void)

  find_available_device_id find a available device ID
- bool check_device_id (int device_id)

  check_device_id check the device is legal
- static bool set_timer (device_management_t ∗new_device)
- static void printf_device (const device_management_t ∗device)
- int create_device (adapter_t ∗adapter)

  create_device create a rs485 device, mount the device to protocol
- int delete_device (int object_id, int device_id)

  delete_device delete a device form device management table.
- int get_device_name (char ∗out, int out_len, int device_id)

  get_device_name get a device name from device database.
- int get_device_type (int device_id)

  get_device_type get a device type from device database, just like air condition, fresh air.....
- int get_device_protocol (int device_id)

  get_device_protocol get a device protocol from device database, just like BACnet, MODUBS...
- int get_device_addr (unsigned char ∗addr, unsigned int addr_len, int device_id)

  get_device_addr get a rs485 device addr, you maybe have no address for some device.
- timer_task_t ∗ get_device_timer (int device_id)

  get_device_timer get a device timer task.
- struct device_profile ∗ get_device_private (int device_id)

  get_device_private get a device private profile
- int get_device_private_numbers (int device_id)

  get_device_private_numbers
- int get_device_object_id (int device_id)

  get_device_object_id get the object id by device id
- int get_device_factory_name (int device_id)

  get_device_factory_name Get the device factory name
- int get_device_retransmission (int device_id)

  get_device_retransmission Get the device retransmission count on bus
- int get_device_timeout_ms (int device_id)

  get_device_timeout_ms Get The device timeout (ms), The bus have send a package have wait timeout count.
- int get_device_address_len (int device_id)

  get_device_address_len Get the device address length.
- device_management_t ∗ get_device_management (int device_id)

  get_device_management get the device management pointer
- int device_managmnt_init (void)

  device_managmnt_init The device management modele have a initialize
- int set_read_device_information (const read_device_return_t ∗info, int device_id)

  set_read_device_information bus have get a device information have wirte it.
- int get_read_device_information (read_device_return_t ∗out, int device_id)

  get_read_device_information It's read a device information called by adapter layer.

Variables

- static device_management_t ∗ glb_device_manage [RS485_DEVICE_MAX_NUMBERS] = { NULL }
- static pthread_mutex_t device_management_lock
- static unsigned char curtain_factory [RS485_CURTAIN_MAX_FACTORY] = { 0 }

7.28.1 Detailed Description

www.enno.com

Date

: Mar 24, 2016

Author

: wong

Definition in file device.c.

7.28.2 Macro Definition Documentation

7.28.2.1 #define RS485_CURTAIN_MAX_FACTORY (16)

Definition at line 43 of file device.c.

7.28.2.2 #define RS485_DEVICE_MAX_NUMBERS (100)

Definition at line 35 of file device.c.

7.28.2.3 #define RS485_DEVICE_NAME_MAX_LENGTH (108)

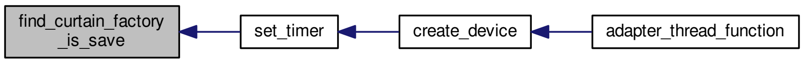
Definition at line 39 of file device.c.

7.28.3 Function Documentation

7.28.3.1 static bool find_curtain_factory_is_save (**rs485_factory_name_enum** ***factory***) `[inline]`, `[static]`

Definition at line 58 of file device.c.

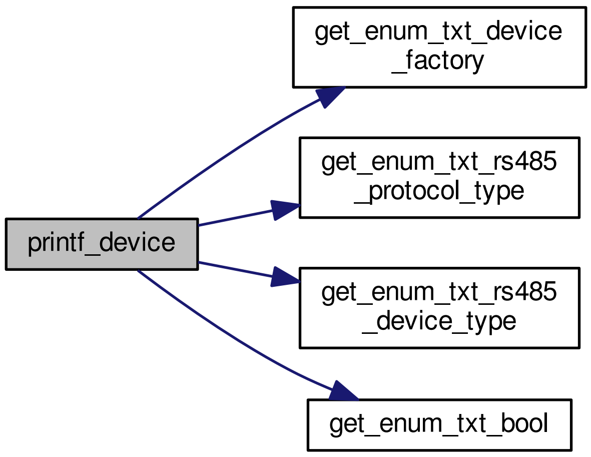
Here is the caller graph for this function:



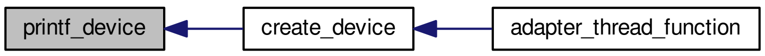
7.28.3.2 static void printf_device (const device_management_t * *device*) [static]

Definition at line 188 of file device.c.

Here is the call graph for this function:


get_enum_txt_device _factory
get_enum_txt_rs485 _protocol_type
printf_device
get_enum_txt_rs485 _device_type
get_enum_txt_bool

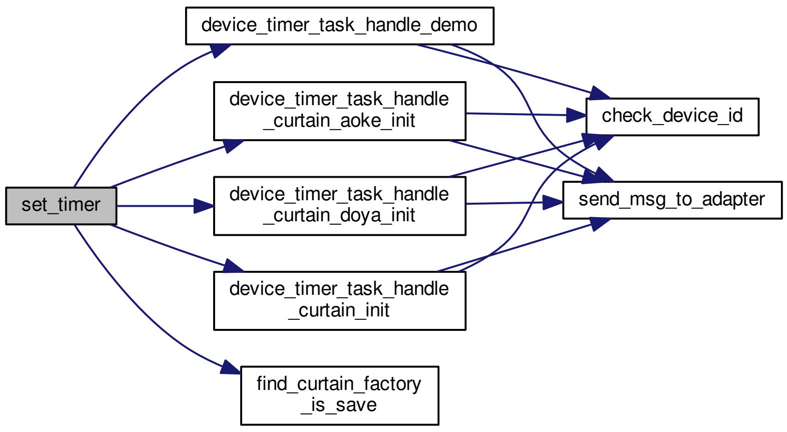
Here is the caller graph for this function:



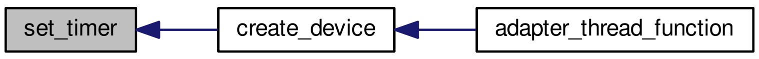
7.28.3.3 static bool set_timer (device_management_t * *new_device*) [inline], [static]

Definition at line 128 of file device.c.

Here is the call graph for this function:


device_timer_task_handle_demo
device_timer_task_handle_curtain_aoke_init
set_timer
device_timer_task_handle_curtain_doya_init
device_timer_task_handle_curtain_init
find_curtain_factory_is_save
check_device_id
send_msg_to_adapter

Here is the caller graph for this function:



7.28.4 Variable Documentation

7.28.4.1 unsigned char curtain_factory[RS485_CURTAIN_MAX_FACTORY] = { 0 } `[static]`

save the curtain factory

Definition at line 54 of file device.c.

7.28.4.2 pthread_mutex_t device_management_lock `[static]`

define the device management lock

Definition at line 51 of file device.c.

7.28.4.3 device_management_t* glb_device_manage[RS485_DEVICE_MAX_NUMBERS] = { NULL } `[static]`

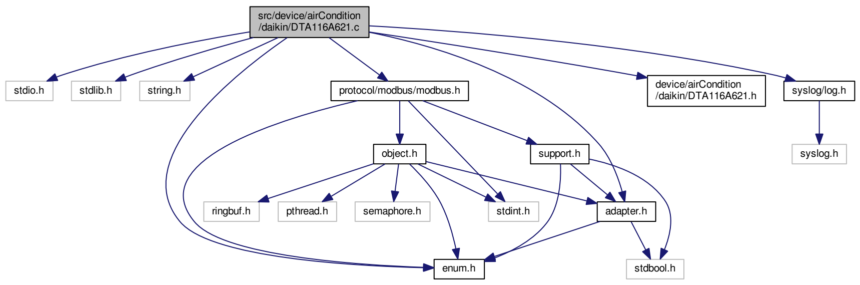
define the device management table

Definition at line 48 of file device.c.

7.29 src/device/airCondition/daikin/DTA116A621.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "enum.h"
#include "adapter.h"
#include "device/airCondition/daikin/DTA116A621.h"
#include "protocol/modbus/modbus.h"
#include "syslog/log.h"
```

Include dependency graph for DTA116A621.c:



Macros

- #define DAIKIN_INPUT_REGISTER_START (30001)
- #define DAIKIN_INPUT_REGISTER_DEVICE_CONNECT_STATUS_START (30002)
- #define DAIKIN_INPUT_REGISTER_DEVICE_CONNECT_STATUS_STOP (30005)
- #define DAIKIN_INPUT_REGISTER_DEVICE_COMM_STATUS_START (30006)
- #define DAIKIN_INPUT_REGISTER_DEVICE_COMM_STATUS_STOP (30009)
- #define DAIKIN_INPUT_REGISTER_DEVICE_FUNCTION_STATUS_START (31001)
- #define DAIKIN_INPUT_REGISTER_DEVICE_FUNCTION_STATUS_STOP (31192)
- #define DAIKIN_INPUT_REGISTER_DEVICE_STATUS_START (32001)
- #define DAIKIN_INPUT_REGISTER_DEVICE_STATUS_STOP (32384)
- #define DAIKIN_HOLD_REGISTER_START (40001)
- #define DAIKIN_HOLD_REGISTER_DEVICE_CONTROL_START (42001)
- #define DAIKIN_HOLD_REGISTER_DEVICE_CONTROL_STOP (42192)
- #define DAIKIN_HOLD_REGISTER_DEVICE_ON_OFF_SWING_FAN_OFFSET (0)
- #define DAIKIN_HOLD_REGISTER_DEVICE_MODE_OFFSET (1)
- #define DAIKIN_HOLD_REGISTER_DEVICE_TEMPERATURE_OFFSET (2)

Functions

- static uint16_t get_daikin_write_register_addr (unsigned char device_addr)

  get_daikin_write_register_addr
- static uint16_t get_daikin_hold_register_value (rs485_device_method_enum method, int value, unsigned char device_addr)
- int daikin_dta116a621_set_temperature (volatile void ∗arg)

  daikin_dta116a621_set_temperature set daikin air condition temperature send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_mode (volatile void ∗arg)

  daikin_dta116a621_set_mode set daikin air conditon mode send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_swing (volatile void ∗arg)

daikin_dta116a621_set_swing set daikin air conditon swing send package to "modbus_port_handle_t"

- int daikin_dta116a621_set_fan (volatile void ∗arg)

    daikin_dta116a621_set_fan set daikin air conditon fan send package to "modbus_port_handle_t"
- int daikin_dta116a621_set_switch (volatile void ∗arg)

    daikin_dta116a621_set_switch set daikin air conditon switch send package to "modbus_port_handle_t"
- int daikin_dta116a621_get_device_info_send (volatile void ∗arg)

    daikin_dta116a621_get_device_info_send set daikin air conditon device information send package to "modbus_↵ port_handle_t"
- int daikin_dta116a621_get_device_info_handle (volatile void ∗arg)

    daikin_dta116a621_get_device_info_handle process daikin air conditon get device information send package to "modbus_port_handle_t"

Variables

- static uint16_t glb_daikin_hold_register_value [64][3]

7.29.1 Macro Definition Documentation

7.29.1.1 #define DAIKIN_HOLD_REGISTER_DEVICE_CONTROL_START (42001)

Definition at line 41 of file DTA116A621.c.

7.29.1.2 #define DAIKIN_HOLD_REGISTER_DEVICE_CONTROL_STOP (42192)

Definition at line 42 of file DTA116A621.c.

7.29.1.3 #define DAIKIN_HOLD_REGISTER_DEVICE_MODE_OFFSET (1)

Definition at line 45 of file DTA116A621.c.

7.29.1.4 #define DAIKIN_HOLD_REGISTER_DEVICE_ON_OFF_SWING_FAN_OFFSET (0)

Definition at line 44 of file DTA116A621.c.

7.29.1.5 #define DAIKIN_HOLD_REGISTER_DEVICE_TEMPERATURE_OFFSET (2)

Definition at line 46 of file DTA116A621.c.

7.29.1.6 #define DAIKIN_HOLD_REGISTER_START (40001)

Definition at line 40 of file DTA116A621.c.

7.29.1.7 #define DAIKIN_INPUT_REGISTER_DEVICE_COMM_STATUS_START (30006)

Definition at line 33 of file DTA116A621.c.

7.29.1.8 #define DAIKIN_INPUT_REGISTER_DEVICE_COMM_STATUS_STOP (30009)

Definition at line 34 of file DTA116A621.c.

7.29.1.9 #define DAIKIN_INPUT_REGISTER_DEVICE_CONNECT_STATUS_START (30002)

Definition at line 31 of file DTA116A621.c.

7.29.1.10 #define DAIKIN_INPUT_REGISTER_DEVICE_CONNECT_STATUS_STOP (30005)

Definition at line 32 of file DTA116A621.c.

7.29.1.11 #define DAIKIN_INPUT_REGISTER_DEVICE_FUNCTION_STATUS_START (31001)

Definition at line 35 of file DTA116A621.c.

7.29.1.12 #define DAIKIN_INPUT_REGISTER_DEVICE_FUNCTION_STATUS_STOP (31192)

Definition at line 36 of file DTA116A621.c.

7.29.1.13 #define DAIKIN_INPUT_REGISTER_DEVICE_STATUS_START (32001)

Definition at line 37 of file DTA116A621.c.

7.29.1.14 #define DAIKIN_INPUT_REGISTER_DEVICE_STATUS_STOP (32384)

Definition at line 38 of file DTA116A621.c.

7.29.1.15 #define DAIKIN_INPUT_REGISTER_START (30001)

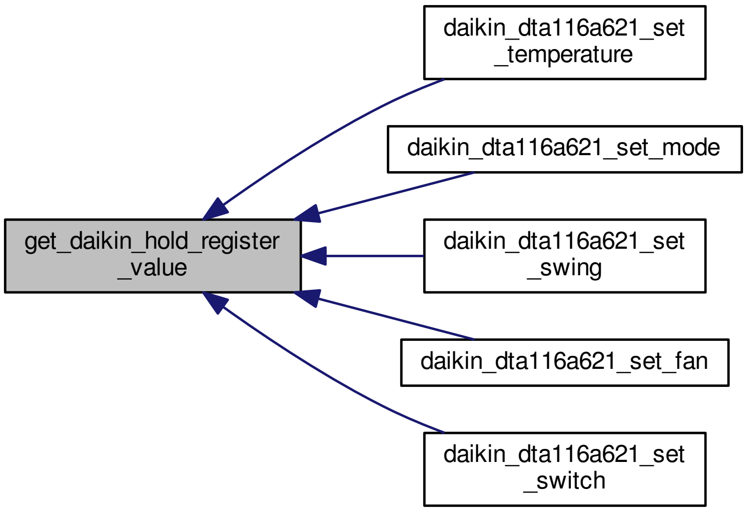
Definition at line 30 of file DTA116A621.c.

7.29.2 Function Documentation

7.29.2.1 static uint16_t get_daikin_hold_register_value (rs485_device_method_enum *method,* int *value,* unsigned char *device_addr*) `[static]`

Definition at line 133 of file DTA116A621.c.

Here is the caller graph for this function:


get_daikin_hold_register_value
daikin_dta116a621_set_temperature
daikin_dta116a621_set_mode
daikin_dta116a621_set_swing
daikin_dta116a621_set_fan
daikin_dta116a621_set_switch

7.29.2.2 static uint16_t get_daikin_write_register_addr (unsigned char *device_addr*) [static]

get_daikin_write_register_addr

Parameters

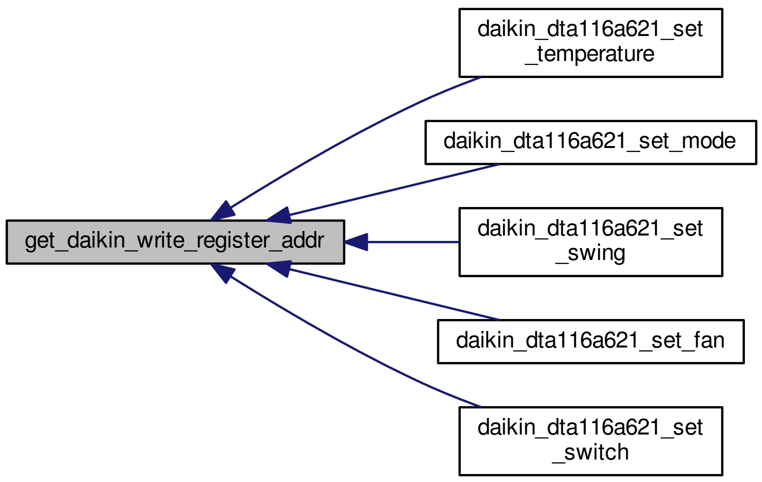
in	*device_addr*	: 0∼63

Returns

The daikin register address

Definition at line 127 of file DTA116A621.c.

Here is the caller graph for this function:


get_daikin_write_register_addr
daikin_dta116a621_set_temperature
daikin_dta116a621_set_mode
daikin_dta116a621_set_swing
daikin_dta116a621_set_fan
daikin_dta116a621_set_switch

7.29.3 Variable Documentation

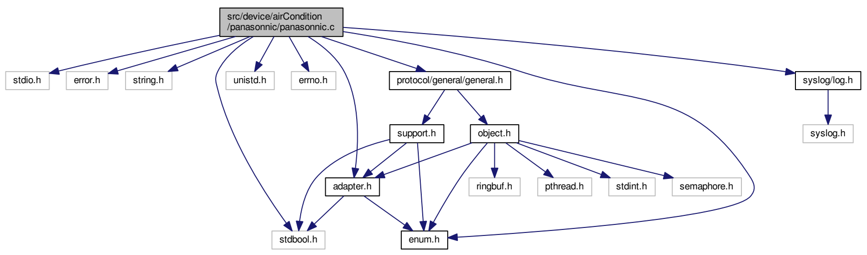
7.29.3.1 uint16_t glb_daikin_hold_register_value[64][3] `[static]`

Definition at line 49 of file DTA116A621.c.

7.30 src/device/airCondition/panasonnic/panasonnic.c File Reference

```
#include <stdio.h>
#include <error.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <errno.h>
#include "enum.h"
#include "adapter.h"
#include "protocol/general/general.h"
#include "syslog/log.h"
```

Include dependency graph for panasonnic.c:



Macros

- #define GET_AIR_ADDR_CID_SEND 0x01
- #define GET_AIR_ADDR
- #define GET_AIR_INFO_CID_SEND 0x02
- #define GET_AIR_INFO_TELECONTROLLER_MODE_AUTO 0x00
- #define GET_AIR_INFO_TELECONTROLLER_MODE_DRY 0x02
- #define GET_AIR_INFO_TELECONTROLLER_MODE_COLD 0x03
- #define GET_AIR_INFO_TELECONTROLLER_MODE_HOT 0x04
- #define GET_AIR_INFO_TELECONTROLLER_MODE_WIND 0x06
- #define GET_AIR_INFO_ON 0x01
- #define GET_AIR_INFO_OFF 0x00
- #define GET_AIR_INFO_SET_WIND_RATE_LOW 0x03
- #define GET_AIR_INFO_SET_WIND_RATE_MIDDLE 0x05
- #define GET_AIR_INFO_SET_WIND_RATE_HIGH 0x07
- #define GET_AIR_INFO_SET_WIND_RATE_MOST 0x09
- #define GET_AIR_INFO_SET_WIND_RATE_AUTO 0x0a
- #define GET_AIR_INFO_SET_WIND_RATE_MUTE 0x0b
- #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL1 0x01
- #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL2 0x02
- #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL3 0x03
- #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL4 0x04
- #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL5 0x05
- #define GET_AIR_INFO_SET_WIND_DIRECTION_FOCUS 0x0e
- #define GET_AIR_INFO_SET_WIND_DIRECTION_WIDE 0x0f
- #define GET_AIR_INFO_SET_TEMPERATURE(x) (x)/2
- #define GET_AIR_INFO_SET_TEMPERATURE_UP 0xe1
- #define GET_AIR_INFO_SET_TEMPERATURE_STD 0xe0
- #define GET_AIR_INFO_SET_TEMPERATURE_DOWN 0xff
- #define GET_AIR_INFO_SET_TEMPERATURE_ADD_5 0xca
- #define GET_AIR_INFO_SET_TEMPERATURE_ADD_1 0xc2
- #define GET_AIR_INFO_SET_TEMPERATURE_DRY 0xc0
- #define GET_AIR_INFO_SET_TEMPERATURE_SUB_1 0xde
- #define GET_AIR_INFO_SET_TEMPERATURE_SUB_5 0xd6
- #define GET_AIR_INFO_TEMPERATURE_UNDERFLOW 0x80
- #define GET_AIR_INFO_TEMPERATURE_OVERFLOW 0x7f
- #define GET_AIR_INFO_TEMPERATURE_UNKNOWN 0x7e
- #define GET_AIR_INFO_ERROR_CODE_NORMAL 0x00
- #define SET_AIR_ARG_CID_SEND 0x03
- #define SET_AIR_ARG_CONFIG_OPT_ALL 0x00
- #define SET_AIR_ARG_CONFIG_OPT_SWITCH 0x01
- #define SET_AIR_ARG_CONFIG_OPT_MODE 0x02
- #define SET_AIR_ARG_CONFIG_OPT_WIND_DIRECTION 0x03
- #define SET_AIR_ARG_CONFIG_OPT_WIND_RATE 0x04
- #define SET_AIR_ARG_CONFIG_OPT_TEMPERATURE 0x05
- #define SET_AIR_ARG_CONFIG_OPT_KEEP 0xff
- #define SET_AIR_ARG_ON 0x00
- #define SET_AIR_ARG_OFF 0x01
- #define SET_AIR_ARG_TELECONTROLLER_MODE_AUTO 0x00
- #define SET_AIR_ARG_TELECONTROLLER_MODE_DRY 0x02
- #define SET_AIR_ARG_TELECONTROLLER_MODE_COLD 0x03
- #define SET_AIR_ARG_TELECONTROLLER_MODE_HOT 0x04
- #define SET_AIR_ARG_TELECONTROLLER_MODE_WIND 0x06
- #define SET_AIR_ARG_WIND_RATE_LOW 0x03
- #define SET_AIR_ARG_WIND_RATE_MIDDLE 0x05

- #define SET_AIR_ARG_WIND_RATE_HIGH 0x07
- #define SET_AIR_ARG_WIND_RATE_MOST 0x09
- #define SET_AIR_ARG_WIND_RATE_AUTO 0x0a
- #define SET_AIR_ARG_WIND_RATE_MUTE 0x0b
- #define SET_AIR_ARG_WIND_DIRECTION_HANDL (0x01 | 0x02 | 0x03 | 0x04 | 0x05)
- #define SET_AIR_ARG_WIND_DIRECTION_FOCUS 0x0e
- #define SET_AIR_ARG_WIND_DIRECTION_WIDE 0x0f
- #define SET_AIR_ARG_TEMPERATURE(x) temperature_to_bin(x)
- #define RESET_AIR_CID_SEND 0x04
- #define SOI_SEND 0xaa
- #define SOI_RECEIVE 0x55
- #define ADR_BROADCAST 0xff
- #define ADR_DEFAULT 0x01
- #define RTN_SEND 0x60
- #define RTN_RECEIVE_CMD_RIGHT 0x01
- #define RTN_RECEIVE_CHK_ERROR 0x02
- #define RTN_RECEIVE_CMD_INVALID 0x03
- #define EOI 0x0d
- #define SEND_ERROR -2
- #define RECEIVE_ERROR -3
- #define RECEIVE_CHK_ERROR -4
- #define RETURN_DATA_INVALID_ERROR -5
- #define RETURN_CHK_ERROR -6
- #define ARG_ERROR -7
- #define UNKNOW_ERROR -8
- #define ERROR -1
- #define PACKAGE_MAX 20
- #define HIGH_CHAR(x) (x)>>4
- #define LOW_CHAR(x) (x) & 0xff

Functions

- static unsigned char calculate_sum_check (unsigned char ∗value, int length)
- static int panasonnic_send_package (unsigned char ∗out, int out_len, unsigned char addr, unsigned char msg_cid, const unsigned char ∗data, int data_len)
- int panasonnic_send_package_handle (volatile void ∗arg)

  panasonnic_send_package_handle The panasonnic package a send buffer interface.
- int panasonnic_recv_package_handle (volatile void ∗arg)

  panasonnic_send_package_handle The panasonnic package a receive buffer processs interface.

Variables

- const unsigned char air_command_table [34][5]

7.30.1 Macro Definition Documentation

7.30.1.1 #define ADR_BROADCAST 0xff

Definition at line 123 of file panasonnic.c.

7.30.1.2 #define ADR_DEFAULT 0x01

Definition at line 124 of file panasonnic.c.

7.30.1.3 #define ARG_ERROR -7

Definition at line 138 of file panasonnic.c.

7.30.1.4 #define EOI 0x0d

Definition at line 129 of file panasonnic.c.

7.30.1.5 #define ERROR -1

define air conditioner transmit frame, it's up RS485 struct Package_air_transmit

{

unsigned char |soi; |

unsigned char |addr; |

unsigned char |cid; |

unsigned char |rtn; |

unsigned char |length; |

length |data; |

unsigned char |sum_chk; |

unsigned char |eoi; |

};

Definition at line 167 of file panasonnic.c.

7.30.1.6 #define GET_AIR_ADDR

Definition at line 39 of file panasonnic.c.

7.30.1.7 #define GET_AIR_ADDR_CID_SEND 0x01

get air conditioner address

Definition at line 38 of file panasonnic.c.

7.30.1.8 #define GET_AIR_INFO_CID_SEND 0x02

get air conditioner information

Definition at line 42 of file panasonnic.c.

7.30.1.9 #define GET_AIR_INFO_ERROR_CODE_NORMAL 0x00

Definition at line 82 of file panasonnic.c.

7.30.1.10 #define GET_AIR_INFO_OFF 0x00

Definition at line 51 of file panasonnic.c.

7.30.1.11 #define GET_AIR_INFO_ON 0x01

Definition at line 50 of file panasonnic.c.

7.30.1.12 #define GET_AIR_INFO_SET_TEMPERATURE(*x*) (x)/2

Definition at line 68 of file panasonnic.c.

7.30.1.13 #define GET_AIR_INFO_SET_TEMPERATURE_ADD_1 0xc2

Definition at line 73 of file panasonnic.c.

7.30.1.14 #define GET_AIR_INFO_SET_TEMPERATURE_ADD_5 0xca

Definition at line 72 of file panasonnic.c.

7.30.1.15 #define GET_AIR_INFO_SET_TEMPERATURE_DOWN 0xff

Definition at line 71 of file panasonnic.c.

7.30.1.16 #define GET_AIR_INFO_SET_TEMPERATURE_DRY 0xc0

Definition at line 74 of file panasonnic.c.

7.30.1.17 #define GET_AIR_INFO_SET_TEMPERATURE_STD 0xe0

Definition at line 70 of file panasonnic.c.

7.30.1.18 #define GET_AIR_INFO_SET_TEMPERATURE_SUB_1 0xde

Definition at line 75 of file panasonnic.c.

7.30.1.19 #define GET_AIR_INFO_SET_TEMPERATURE_SUB_5 0xd6

Definition at line 76 of file panasonnic.c.

7.30.1.20 #define GET_AIR_INFO_SET_TEMPERATURE_UP 0xe1

Definition at line 69 of file panasonnic.c.

7.30.1.21 #define GET_AIR_INFO_SET_WIND_DIRECTION_FOCUS 0x0e

Definition at line 65 of file panasonnic.c.

7.30.1.22 #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL1 0x01

Definition at line 60 of file panasonnic.c.

7.30.1.23 #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL2 0x02

Definition at line 61 of file panasonnic.c.

7.30.1.24 #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL3 0x03

Definition at line 62 of file panasonnic.c.

7.30.1.25 #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL4 0x04

Definition at line 63 of file panasonnic.c.

7.30.1.26 #define GET_AIR_INFO_SET_WIND_DIRECTION_HANDL5 0x05

Definition at line 64 of file panasonnic.c.

7.30.1.27 #define GET_AIR_INFO_SET_WIND_DIRECTION_WIDE 0x0f

Definition at line 66 of file panasonnic.c.

7.30.1.28 #define GET_AIR_INFO_SET_WIND_RATE_AUTO 0x0a

Definition at line 57 of file panasonnic.c.

7.30.1.29 #define GET_AIR_INFO_SET_WIND_RATE_HIGH 0x07

Definition at line 55 of file panasonnic.c.

7.30.1.30 #define GET_AIR_INFO_SET_WIND_RATE_LOW 0x03

Definition at line 53 of file panasonnic.c.

7.30.1.31 #define GET_AIR_INFO_SET_WIND_RATE_MIDDLE 0x05

Definition at line 54 of file panasonnic.c.

7.30.1.32 #define GET_AIR_INFO_SET_WIND_RATE_MOST 0x09

Definition at line 56 of file panasonnic.c.

7.30.1.33 #define GET_AIR_INFO_SET_WIND_RATE_MUTE 0x0b

Definition at line 58 of file panasonnic.c.

7.30.1.34 #define GET_AIR_INFO_TELECONTROLLER_MODE_AUTO 0x00

Definition at line 44 of file panasonnic.c.

7.30.1.35 #define GET_AIR_INFO_TELECONTROLLER_MODE_COLD 0x03

Definition at line 46 of file panasonnic.c.

7.30.1.36 #define GET_AIR_INFO_TELECONTROLLER_MODE_DRY 0x02

Definition at line 45 of file panasonnic.c.

7.30.1.37 #define GET_AIR_INFO_TELECONTROLLER_MODE_HOT 0x04

Definition at line 47 of file panasonnic.c.

7.30.1.38 #define GET_AIR_INFO_TELECONTROLLER_MODE_WIND 0x06

Definition at line 48 of file panasonnic.c.

7.30.1.39 #define GET_AIR_INFO_TEMPERATURE_OVERFLOW 0x7f

Definition at line 79 of file panasonnic.c.

7.30.1.40 #define GET_AIR_INFO_TEMPERATURE_UNDERFLOW 0x80

Definition at line 78 of file panasonnic.c.

7.30.1.41 #define GET_AIR_INFO_TEMPERATURE_UNKNOWN 0x7e

Definition at line 80 of file panasonnic.c.

7.30.1.42 #define HIGH_CHAR(*x*) (x)>>4

Definition at line 172 of file panasonnic.c.

7.30.1.43 #define LOW_CHAR(*x*) (x) & 0xff

Definition at line 173 of file panasonnic.c.

7.30.1.44 #define PACKAGE_MAX 20

Definition at line 170 of file panasonnic.c.

7.30.1.45 #define RECEIVE_CHK_ERROR -4

Definition at line 135 of file panasonnic.c.

7.30.1.46 #define RECEIVE_ERROR -3

Definition at line 134 of file panasonnic.c.

7.30.1.47 #define RESET_AIR_CID_SEND 0x04

when the air conditioner unusual, will reset it

Definition at line 118 of file panasonnic.c.

7.30.1.48 #define RETURN_CHK_ERROR -6

Definition at line 137 of file panasonnic.c.

7.30.1.49 #define RETURN_DATA_INVALID_ERROR -5

Definition at line 136 of file panasonnic.c.

7.30.1.50 #define RTN_RECEIVE_CHK_ERROR 0x02

Definition at line 127 of file panasonnic.c.

7.30.1.51 #define RTN_RECEIVE_CMD_INVALID 0x03

Definition at line 128 of file panasonnic.c.

7.30.1.52 #define RTN_RECEIVE_CMD_RIGHT 0x01

Definition at line 126 of file panasonnic.c.

7.30.1.53 #define RTN_SEND 0x60

Definition at line 125 of file panasonnic.c.

7.30.1.54 #define SEND_ERROR -2

Definition at line 133 of file panasonnic.c.

7.30.1.55 #define SET_AIR_ARG_CID_SEND 0x03

set air conditioner arguments

Definition at line 85 of file panasonnic.c.

7.30.1.56 #define SET_AIR_ARG_CONFIG_OPT_ALL 0x00

Definition at line 87 of file panasonnic.c.

7.30.1.57 #define SET_AIR_ARG_CONFIG_OPT_KEEP 0xff

Definition at line 93 of file panasonnic.c.

7.30.1.58 #define SET_AIR_ARG_CONFIG_OPT_MODE 0x02

Definition at line 89 of file panasonnic.c.

7.30.1.59 #define SET_AIR_ARG_CONFIG_OPT_SWITCH 0x01

Definition at line 88 of file panasonnic.c.

7.30.1.60 #define SET_AIR_ARG_CONFIG_OPT_TEMPERATURE 0x05

Definition at line 92 of file panasonnic.c.

7.30.1.61 #define SET_AIR_ARG_CONFIG_OPT_WIND_DIRECTION 0x03

Definition at line 90 of file panasonnic.c.

7.30.1.62 #define SET_AIR_ARG_CONFIG_OPT_WIND_RATE 0x04

Definition at line 91 of file panasonnic.c.

7.30.1.63 #define SET_AIR_ARG_OFF 0x01

Definition at line 96 of file panasonnic.c.

7.30.1.64 #define SET_AIR_ARG_ON 0x00

Definition at line 95 of file panasonnic.c.

7.30.1.65 #define SET_AIR_ARG_TELECONTROLLER_MODE_AUTO 0x00

Definition at line 98 of file panasonnic.c.

7.30.1.66 #define SET_AIR_ARG_TELECONTROLLER_MODE_COLD 0x03

Definition at line 100 of file panasonnic.c.

7.30.1.67 #define SET_AIR_ARG_TELECONTROLLER_MODE_DRY 0x02

Definition at line 99 of file panasonnic.c.

7.30.1.68 #define SET_AIR_ARG_TELECONTROLLER_MODE_HOT 0x04

Definition at line 101 of file panasonnic.c.

7.30.1.69 #define SET_AIR_ARG_TELECONTROLLER_MODE_WIND 0x06

Definition at line 102 of file panasonnic.c.

7.30.1.70 #define SET_AIR_ARG_TEMPERATURE(*x*) temperature_to_bin(x)

Definition at line 115 of file panasonnic.c.

7.30.1.71 #define SET_AIR_ARG_WIND_DIRECTION_FOCUS 0x0e

Definition at line 112 of file panasonnic.c.

7.30.1.72 #define SET_AIR_ARG_WIND_DIRECTION_HANDL (0x01 | 0x02 | 0x03 | 0x04 | 0x05)

Definition at line 111 of file panasonnic.c.

7.30.1.73 #define SET_AIR_ARG_WIND_DIRECTION_WIDE 0x0f

Definition at line 113 of file panasonnic.c.

7.30.1.74 #define SET_AIR_ARG_WIND_RATE_AUTO 0x0a

Definition at line 108 of file panasonnic.c.

7.30.1.75 #define SET_AIR_ARG_WIND_RATE_HIGH 0x07

Definition at line 106 of file panasonnic.c.

7.30.1.76 #define SET_AIR_ARG_WIND_RATE_LOW 0x03

Definition at line 104 of file panasonnic.c.

7.30.1.77 #define SET_AIR_ARG_WIND_RATE_MIDDLE 0x05

Definition at line 105 of file panasonnic.c.

7.30.1.78 #define SET_AIR_ARG_WIND_RATE_MOST 0x09

Definition at line 107 of file panasonnic.c.

7.30.1.79 #define SET_AIR_ARG_WIND_RATE_MUTE 0x0b

Definition at line 109 of file panasonnic.c.

7.30.1.80 #define SOI_RECEIVE 0x55

Definition at line 122 of file panasonnic.c.

7.30.1.81 #define SOI_SEND 0xaa

package protocol

Definition at line 121 of file panasonnic.c.

7.30.1.82 #define UNKNOW_ERROR -8

Definition at line 139 of file panasonnic.c.

7.30.2 Function Documentation

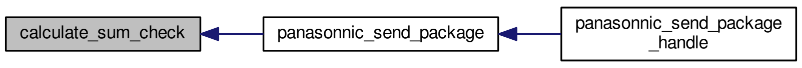
7.30.2.1 static unsigned char calculate_sum_check (unsigned char ∗ *value,* int *length*) [static]

static function declare

static function define

Definition at line 279 of file panasonnic.c.

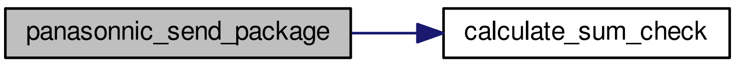
Here is the caller graph for this function:



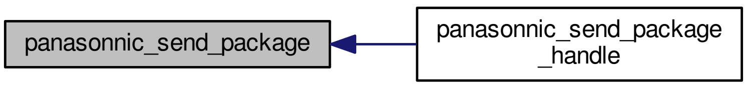
7.30.2.2 static int panasonnic_send_package (unsigned char ∗ *out,* int *out_len,* unsigned char *addr,* unsigned char *msg_cid,* const unsigned char ∗ *data,* int *data_len*) [static]

Definition at line 676 of file panasonnic.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.30.3 Variable Documentation

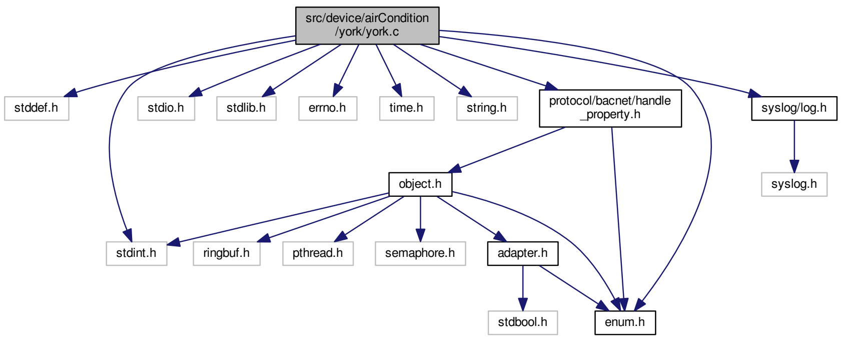
7.30.3.1 const unsigned char air_command_table[34][5]

Definition at line 218 of file panasonnic.c.

7.31 src/device/airCondition/york/york.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include <string.h>
#include "protocol/bacnet/handle_property.h"
#include "enum.h"
#include "syslog/log.h"
```

Include dependency graph for york.c:



Enumerations

- enum york_air_conditioner_object {
  AI_OUTDOOR_TEMPERATURE, AI_INDOOR_TEMPERATURE, AI_SET_TEMPERATURE, AI_INDOOR←_HUMIDITY,
  AI_SET_HUMIDITY, AI_COMPRESSOR_FREQUENCY, AI_NET_CLEAR_TIME_LEFT, BI_ON_OFF_ST←ATUS,
  BI_ERROR_RESET_STATUS, BI_NET_CLEAR_RESET_STATUS, BI_SLEEP_STATUS, BI_ELECTRIC←AL_HEAT_STATUS,
  BI_HEALTH_AIR_STATUS, BI_HOT_WATER_STATUS, BI_NEW_AIR_STATUS, BI_FIX_RUN_STATUS,
  BI_SAVING_STATUS, BI_DEFROST, BI_COMPRESSOR_RUNNING_STATUS, BI_COOL_ONLY_STAT←US,
  BI_CENTRAL_CONONLY_STATUS, MI_RESERVED, MI_MODE, MI_FAN,
  MI_SWING, MI_VENTILATION, MI_LOCAL_SET, MI_ERROR,
  MI_COMMUNICATION_STATUS, MI_INDOOR_STYLE, AV_SET_TEMPERATURE, AV_SET_HUMIDUTY
  }

Functions

- int get_air_york_write_args (bacnet_write_args_t ∗args, unsigned int device_id, int command, char ∗value)
- int get_air_york_read_args (bacnet_read_args_t ∗args, unsigned int device_id)

get_air_york_read_args The york air confition bacnet read interface
- int get_air_york_instance (unsigned char mac)

  get_air_york_instance get the youk bacnet instance.

Variables

- static const int york_air_condition_object [ ][7]
- static const int york_air_condition_read_property [ ][3]

7.31.1 Enumeration Type Documentation

7.31.1.1 enum york_air_conditioner_object

Enumerator

AI_OUTDOOR_TEMPERATURE

AI_INDOOR_TEMPERATURE

AI_SET_TEMPERATURE

AI_INDOOR_HUMIDITY

AI_SET_HUMIDITY

AI_COMPRESSOR_FREQUENCY

AI_NET_CLEAR_TIME_LEFT

BI_ON_OFF_STATUS

BI_ERROR_RESET_STATUS

BI_NET_CLEAR_RESET_STATUS

BI_SLEEP_STATUS

BI_ELECTRICAL_HEAT_STATUS

BI_HEALTH_AIR_STATUS

BI_HOT_WATER_STATUS

BI_NEW_AIR_STATUS

BI_FIX_RUN_STATUS

BI_SAVING_STATUS

BI_DEFROST

BI_COMPRESSOR_RUNNING_STATUS

BI_COOL_ONLY_STATUS

BI_CENTRAL_CONONLY_STATUS

MI_RESERVED

MI_MODE

MI_FAN

MI_SWING

MI_VENTILATION

MI_LOCAL_SET

MI_ERROR

MI_COMMUNICATION_STATUS

MI_INDOOR_STYLE

AV_SET_TEMPERATURE

AV_SET_HUMIDUTY

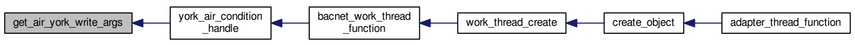
Definition at line 21 of file york.c.

7.31.2 Function Documentation

7.31.2.1 int get_air_york_write_args (bacnet_write_args_t * *args,* unsigned int *device_id,* int *command,* char * *value*)

Definition at line 161 of file york.c.

Here is the caller graph for this function:



7.31.3 Variable Documentation

7.31.3.1 const int york_air_condition_object[ ][7] [static]

Definition at line 61 of file york.c.

7.31.3.2 const int york_air_condition_read_property[ ][3] [static]

Initial value:

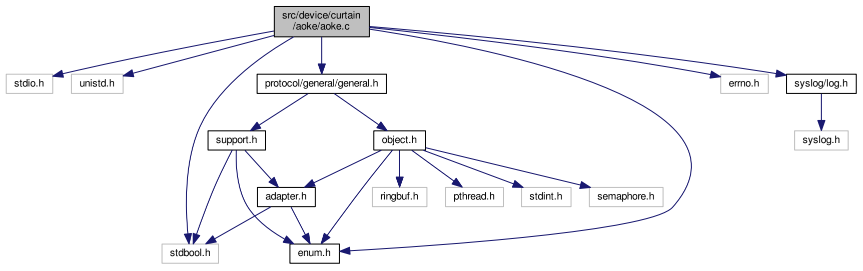
```
=
{
    {0, 0, 85},
    {0, 1, 85},
    {0, 2, 85},
    {0, 3, 85},
    {0, 4, 85},
    {0, 5, 85},
    {3, 1, 85},
}
```

Definition at line 140 of file york.c.

7.32 src/device/curtain/aoke/aoke.c File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <stdbool.h>
#include <errno.h>
#include "enum.h"
#include "syslog/log.h"
#include "protocol/general/general.h"
```

Include dependency graph for aoke.c:



Data Structures

- struct rs485_curtain_ao_ke_send_package_t

Macros

- #define RS485_CURTAIN_AO_KE_SI 0x9a
- #define RS485_CURTAIN_AO_KE_COMMAND_CONTROL 0x0a
- #define RS485_CURTAIN_AO_KE_COMMAND_POSTION 0xdd
- #define RS485_CURTAIN_AO_KE_COMMAND_SETTING 0xd5
- #define RS485_CURTAIN_AO_KE_COMMAND_GETTING 0xcc
- #define RS485_CURTAIN_AO_KE_COMMAND_ADDRING 0xaa
- #define RS485_CURTAIN_AO_KE_COMMAND_DELETE 0xa6
- #define RS485_CURTAIN_AO_KE_COMMAND_POINT 0xda
- #define RS485_CURTAIN_AO_KE_CONTROL_UP 0xdd
- #define RS485_CURTAIN_AO_KE_CONTROL_STOP 0x0d
- #define RS485_CURTAIN_AO_KE_CONTROL_DOWN 0xee
- #define RS485_CURTAIN_AO_KE_CONTROL_SET_ADDR 0xaa
- #define RS485_CURTAIN_AO_KE_CONTROL_DELETE_ADDR 0xa6
- #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_1 0x01
- #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_2 0x02
- #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_3 0x03
- #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_4 0x04
- #define RS485_CURTIAN_AO_KE_POSTION 0x00
- #define RS485_CURTIAN_AO_KE_SETTING_HANDLE 0x01
- #define RS485_CURTIAN_AO_KE_GETTING_STATUS 0xcc
- #define RS485_CURTAIN_AO_KE_ADDRING 0xaa
- #define RS485_CURTAIN_AO_KE_DELETE 0xa6
- #define RS485_CURTAIN_AO_KE_POINT_UP 0xdd
- #define RS485_CURTAIN_AO_KE_POINT_MIDDLE 0xcc
- #define RS485_CURTAIN_AO_KE_POINT_DOWN 0xda
- #define RS485_CURTAIN_AO_KE_POINT_SAVE 0xaa
- #define RS485_CURTAIN_AO_KE_POINT_DELETE 0x00

Functions

- static unsigned char get_check (rs485_curtain_ao_ke_send_package_t ∗package)
- int aoke_send_package_handle (volatile void ∗arg)

  aoke_send_package_handle aoke curtian package a send buffer
- int aoke_recv_package_handle (volatile void ∗arg)

  aoke_recv_package_handle aoke curtain process the receive package

7.32.1 Macro Definition Documentation

7.32.1.1 #define RS485_CURTAIN_AO_KE_ADDRING 0xaa

Definition at line 74 of file aoke.c.

7.32.1.2 #define RS485_CURTAIN_AO_KE_COMMAND_ADDRING 0xaa

Definition at line 52 of file aoke.c.

7.32.1.3 #define RS485_CURTAIN_AO_KE_COMMAND_CONTROL 0x0a

Definition at line 48 of file aoke.c.

7.32.1.4 #define RS485_CURTAIN_AO_KE_COMMAND_DELETE 0xa6

Definition at line 53 of file aoke.c.

7.32.1.5 #define RS485_CURTAIN_AO_KE_COMMAND_GETTING 0xcc

Definition at line 51 of file aoke.c.

7.32.1.6 #define RS485_CURTAIN_AO_KE_COMMAND_POINT 0xda

Definition at line 54 of file aoke.c.

7.32.1.7 #define RS485_CURTAIN_AO_KE_COMMAND_POSTION 0xdd

Definition at line 49 of file aoke.c.

7.32.1.8 #define RS485_CURTAIN_AO_KE_COMMAND_SETTING 0xd5

Definition at line 50 of file aoke.c.

7.32.1.9 #define RS485_CURTAIN_AO_KE_CONTROL_DELETE_ADDR 0xa6

Definition at line 61 of file aoke.c.

7.32.1.10 #define RS485_CURTAIN_AO_KE_CONTROL_DOWN 0xee

Definition at line 59 of file aoke.c.

7.32.1.11 #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_1 0x01

Definition at line 62 of file aoke.c.

7.32.1.12 #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_2 0x02

Definition at line 63 of file aoke.c.

7.32.1.13 #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_3 0x03

Definition at line 64 of file aoke.c.

7.32.1.14 #define RS485_CURTAIN_AO_KE_CONTROL_MIDDLE_4 0x04

Definition at line 65 of file aoke.c.

7.32.1.15 #define RS485_CURTAIN_AO_KE_CONTROL_SET_ADDR 0xaa

Definition at line 60 of file aoke.c.

7.32.1.16 #define RS485_CURTAIN_AO_KE_CONTROL_STOP 0x0d

Definition at line 58 of file aoke.c.

7.32.1.17 #define RS485_CURTAIN_AO_KE_CONTROL_UP 0xdd

Definition at line 57 of file aoke.c.

7.32.1.18 #define RS485_CURTAIN_AO_KE_DELETE 0xa6

Definition at line 76 of file aoke.c.

7.32.1.19 #define RS485_CURTAIN_AO_KE_POINT_DELETE 0x00

Definition at line 82 of file aoke.c.

7.32.1.20 #define RS485_CURTAIN_AO_KE_POINT_DOWN 0xda

Definition at line 80 of file aoke.c.

7.32.1.21 #define RS485_CURTAIN_AO_KE_POINT_MIDDLE 0xcc

Definition at line 79 of file aoke.c.

7.32.1.22 #define RS485_CURTAIN_AO_KE_POINT_SAVE 0xaa

Definition at line 81 of file aoke.c.

7.32.1.23 #define RS485_CURTAIN_AO_KE_POINT_UP 0xdd

Definition at line 78 of file aoke.c.

7.32.1.24 #define RS485_CURTAIN_AO_KE_SI 0x9a

ao_ke curtain on RS485 baud rate : 9600 stop bit : 1 data bit : 8 parity bit : NULL

format:

|**SI** |**ADDR0** |**ADDR1** |**ADDR2** |**CMD** |**DATA** |**CHECK**|

Definition at line 45 of file aoke.c.

7.32.1.25 #define RS485_CURTIAN_AO_KE_GETTING_STATUS 0xcc

Definition at line 72 of file aoke.c.

7.32.1.26 #define RS485_CURTIAN_AO_KE_POSTION 0x00

Definition at line 68 of file aoke.c.

7.32.1.27 #define RS485_CURTIAN_AO_KE_SETTING_HANDLE 0x01

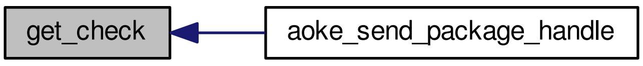
Definition at line 70 of file aoke.c.

7.32.2 Function Documentation

7.32.2.1 static unsigned char get_check (rs485_curtain_ao_ke_send_package_t * *package*) `[static]`

Definition at line 113 of file aoke.c.

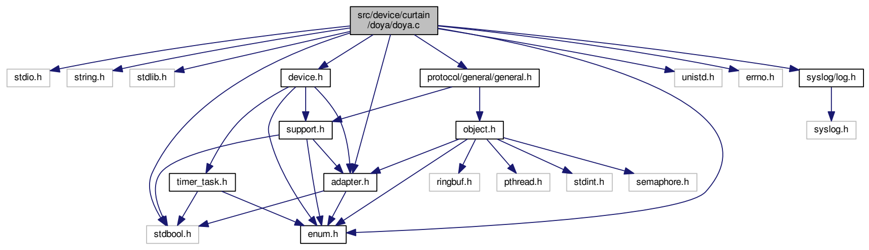
Here is the caller graph for this function:



7.33 src/device/curtain/doya/doya.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <errno.h>
#include "enum.h"
#include "syslog/log.h"
#include "adapter.h"
#include "device.h"
#include "protocol/general/general.h"
```

Include dependency graph for doya.c:



Data Structures

- struct package

Macros

- #define CURTAIN_COMMAND 0x03
- #define CURTAIN_READ 0x01
- #define CURTAIN_WRITE 0x02
- #define CURTAIN_COMMAND_OPEN 0x01
- #define CURTAIN_COMMAND_CLOSE 0x02
- #define CURTAIN_COMMAND_STOP 0x03
- #define CURTAIN_COMMAND_PERCENT 0x04
- #define CURTAIN_COMMAND_DELETE 0x07
- #define CURTAIN_COMMAND_REFACTORY 0x08
- #define CURTAIN_READ_WRITE_ADDR_LOW 0x00
- #define CURTAIN_READ_WRITE_ADDR_HIGH 0x01
- #define CURTAIN_READ_WRITE_ADDR 0x00
- #define CURTAIN_READ_WRITE_PERCENT 0x02
- #define CURTAIN_READ_WRITE_DIRECTION 0x03
- #define CURTAIN_READ_WRITE_HANDLE 0x04
- #define CURTAIN_READ_WRITE_SWITCH_PASSIVE 0x27
- #define CURTAIN_READ_WRITE_SWITCH_ACTIVE 0x28
- #define CURTAIN_READ_WRITE_VERSION 0xfe
- #define MOTOR_POSITIVE 0x00
- #define MOTOR_NEGATIVE 0x01
- #define HANDLE_ENABLE 0x00
- #define HANDLE_DISABLE 0x01
- #define SWITCH_PASSIVE_DOUBLE_REBOUND 0x01
- #define SWITCH_PASSIVE_DOUBLE_NO_REBOUND 0x02
- #define SWITCH_PASSIVE_ELECTRONIC_DC246 0x03
- #define SWITCH_PASSIVE_SINGLE_CYCLE 0x04
- #define SWITCH_ACTIVE_DOUBLE_LINE 0x00
- #define SWITCH_ACTIVE_SINGLE_LINE 0x01

Enumerations

- enum command {
  COMMAND_OPEN, COMMAND_CLOSE, COMMAND_STOP, COMMAND_PERCENT,
  COMMAND_DELETE, COMMAND_REFACTORY, WO_ADDR, RO_PERCENT,
  RW_DIRECTION, RW_HANDLE, RW_SWITCH_PASSIVE, RW_SWITCH_ACTIVE,
  RO_VERSION }

Functions

- static int rs485_send (int port, void ∗buffer, int len)
- static int rs485_read (int port, void ∗buffer, int len)
- static void modbus_crc16 (unsigned char result[2], const unsigned char ∗pucFrame, int usLen)
- static int send_package (int port, const struct package ∗package)
- int receive_package (int port, struct package ∗package)
- int open_curtain_no_reply (int port, const unsigned char addr[2])
- int close_curtain_no_reply (int port, const unsigned char addr[2])
- int stop_curtain_no_reply (int port, const unsigned char addr[2])
- int percent_curtain_no_reply (int port, const unsigned char addr[2], unsigned char data)
- int delete_track_curtain_no_reply (int port, const unsigned char addr[2])
- int refactory_curtain_no_reply (int port, const unsigned char addr[2])
- int set_addr_curtain_no_reply (int port, const unsigned char addr[2], unsigned char set[2])
- int set_direction_curtain_no_reply (int port, const unsigned char addr[2], unsigned char same)
- int set_handle_enable_curtain_no_reply (int port, const unsigned char addr[2], unsigned char enable)
- int set_switch_passive_curtain_no_reply (int port, const unsigned char addr[2], unsigned char type)
- int set_switch_active_curtain_no_reply (int port, const unsigned char addr[2], unsigned char type)
- int read_percent_curtain_no_reply (int port, const unsigned char addr[2])
- int read_version_curtain_no_reply (int port, const unsigned char addr[2])
- static int doya_send_package (unsigned char ∗send_buffer, int buffer_len, const struct package ∗package)
- int doya_send_package_handle (volatile void ∗arg)

  doya_send_package_handle The dooya curtain package a send buffer
- int doya_recv_package_handle (volatile void ∗arg)

  doya_recv_package_handle The dooya curtain process the receive data.

Variables

- static const unsigned char crc_high [ ]
- static const unsigned char crc_low [ ]

7.33.1 Macro Definition Documentation

7.33.1.1 #define CURTAIN_COMMAND 0x03

Definition at line 90 of file doya.c.

7.33.1.2 #define CURTAIN_COMMAND_CLOSE 0x02

Definition at line 96 of file doya.c.

7.33.1.3 #define CURTAIN_COMMAND_DELETE 0x07

Definition at line 99 of file doya.c.

7.33.1.4 #define CURTAIN_COMMAND_OPEN 0x01

Definition at line 95 of file doya.c.

7.33.1.5 #define CURTAIN_COMMAND_PERCENT 0x04

Definition at line 98 of file doya.c.

7.33.1.6 #define CURTAIN_COMMAND_REFACTORY 0x08

Definition at line 100 of file doya.c.

7.33.1.7 #define CURTAIN_COMMAND_STOP 0x03

Definition at line 97 of file doya.c.

7.33.1.8 #define CURTAIN_READ 0x01

Definition at line 91 of file doya.c.

7.33.1.9 #define CURTAIN_READ_WRITE_ADDR 0x00

Definition at line 105 of file doya.c.

7.33.1.10 #define CURTAIN_READ_WRITE_ADDR_HIGH 0x01

Definition at line 104 of file doya.c.

7.33.1.11 #define CURTAIN_READ_WRITE_ADDR_LOW 0x00

Definition at line 103 of file doya.c.

7.33.1.12 #define CURTAIN_READ_WRITE_DIRECTION 0x03

Definition at line 107 of file doya.c.

7.33.1.13 #define CURTAIN_READ_WRITE_HANDLE 0x04

Definition at line 108 of file doya.c.

7.33.1.14 #define CURTAIN_READ_WRITE_PERCENT 0x02

Definition at line 106 of file doya.c.

7.33.1.15 #define CURTAIN_READ_WRITE_SWITCH_ACTIVE 0x28

Definition at line 110 of file doya.c.

7.33.1.16 #define CURTAIN_READ_WRITE_SWITCH_PASSIVE 0x27

Definition at line 109 of file doya.c.

7.33.1.17 #define CURTAIN_READ_WRITE_VERSION 0xfe

Definition at line 111 of file doya.c.

7.33.1.18 #define CURTAIN_WRITE 0x02

Definition at line 92 of file doya.c.

7.33.1.19 #define HANDLE_DISABLE 0x01

Definition at line 118 of file doya.c.

7.33.1.20 #define HANDLE_ENABLE 0x00

Definition at line 117 of file doya.c.

7.33.1.21 #define MOTOR_NEGATIVE 0x01

Definition at line 115 of file doya.c.

7.33.1.22 #define MOTOR_POSITIVE 0x00

Definition at line 114 of file doya.c.

7.33.1.23 #define SWITCH_ACTIVE_DOUBLE_LINE 0x00

Definition at line 125 of file doya.c.

7.33.1.24 #define SWITCH_ACTIVE_SINGLE_LINE 0x01

Definition at line 126 of file doya.c.

7.33.1.25 #define SWITCH_PASSIVE_DOUBLE_NO_REBOUND 0x02

Definition at line 121 of file doya.c.

7.33.1.26 #define SWITCH_PASSIVE_DOUBLE_REBOUND 0x01

Definition at line 120 of file doya.c.

7.33.1.27 #define SWITCH_PASSIVE_ELECTRONIC_DC246 0x03

Definition at line 122 of file doya.c.

7.33.1.28 #define SWITCH_PASSIVE_SINGLE_CYCLE 0x04

Definition at line 123 of file doya.c.

7.33.2 Enumeration Type Documentation

7.33.2.1 enum command

du_ya curtain on RS485 baud rate : 9600 stop bit : 1 data bit : 8 parity bit : NULL

Enumerator

- ***COMMAND_OPEN***
- ***COMMAND_CLOSE***
- ***COMMAND_STOP***
- ***COMMAND_PERCENT***
- ***COMMAND_DELETE***
- ***COMMAND_REFACTORY***
- ***WO_ADDR***
- ***RO_PERCENT***
- ***RW_DIRECTION***
- ***RW_HANDLE***
- ***RW_SWITCH_PASSIVE***
- ***RW_SWITCH_ACTIVE***
- ***RO_VERSION***

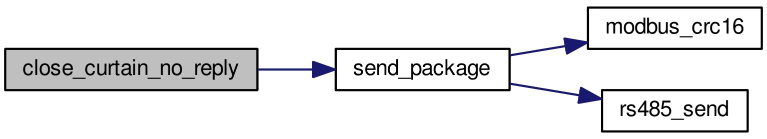
Definition at line 52 of file doya.c.

7.33.3 Function Documentation

7.33.3.1 int close_curtain_no_reply (int *port,* const unsigned char *addr[2]*)

Definition at line 389 of file doya.c.

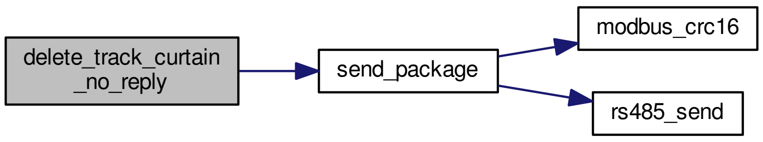
Here is the call graph for this function:



7.33.3.2 int delete_track_curtain_no_reply (int *port,* const unsigned char *addr[2]*)

Definition at line 442 of file doya.c.

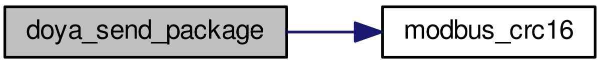
Here is the call graph for this function:



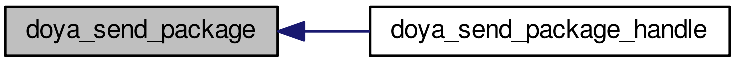
7.33.3.3 static int doya_send_package (unsigned char ∗ *send_buffer,* int *buffer_len,* const struct package ∗ *package*) [static]

Definition at line 607 of file doya.c.

Here is the call graph for this function:



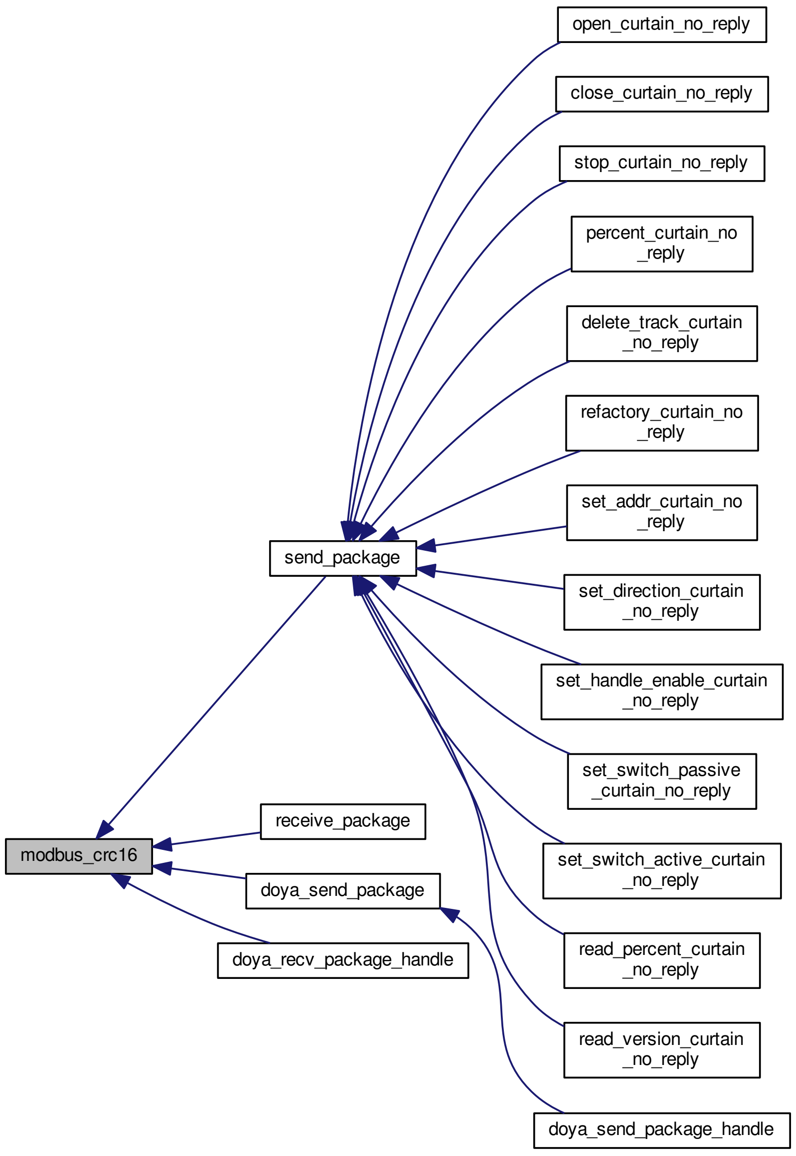
Here is the caller graph for this function:



7.33.3.4 static void modbus_crc16 (unsigned char *result[2],* const unsigned char ∗ *pucFrame,* int *usLen*) [static]

Definition at line 192 of file doya.c.

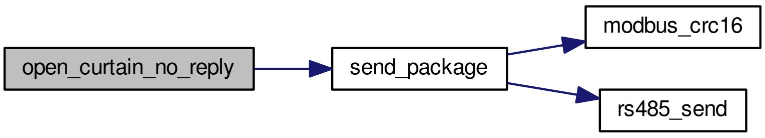
Here is the caller graph for this function:


open_curtain_no_reply
close_curtain_no_reply
stop_curtain_no_reply
percent_curtain_no_reply
delete_track_curtain_no_reply
refactory_curtain_no_reply
set_addr_curtain_no_reply
set_direction_curtain_no_reply
set_handle_enable_curtain_no_reply
set_switch_passive_curtain_no_reply
set_switch_active_curtain_no_reply
read_percent_curtain_no_reply
read_version_curtain_no_reply
doya_send_package_handle
send_package
modbus_crc16
receive_package
doya_send_package
doya_recv_package_handle

7.33.3.5 int open_curtain_no_reply (int *port,* const unsigned char *addr[2]*)

Definition at line 372 of file doya.c.

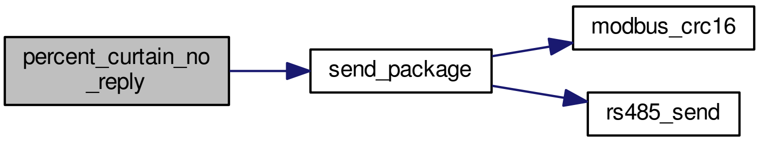
Here is the call graph for this function:



7.33.3.6 int percent_curtain_no_reply (int *port,* const unsigned char *addr[2],* unsigned char *data*)

Definition at line 423 of file doya.c.

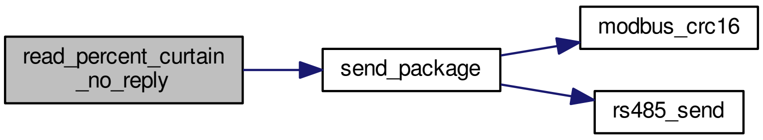
Here is the call graph for this function:



7.33.3.7 int read_percent_curtain_no_reply (int *port,* const unsigned char *addr[2]*)

Definition at line 572 of file doya.c.

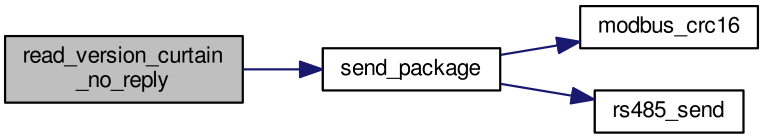
Here is the call graph for this function:



7.33.3.8 int read_version_curtain_no_reply (int *port,* const unsigned char *addr[2]*)

Definition at line 589 of file doya.c.

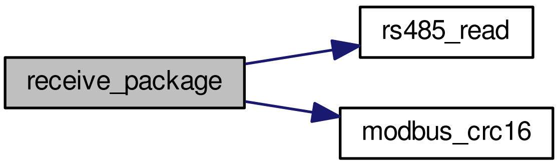
Here is the call graph for this function:



7.33.3.9 int receive_package (int *port*, struct package ∗ *package*)

Definition at line 311 of file doya.c.

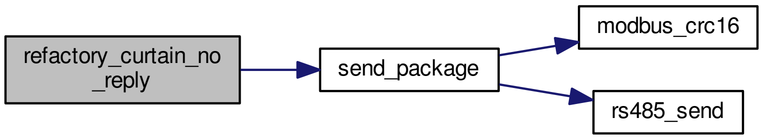
Here is the call graph for this function:



7.33.3.10 int refactory_curtain_no_reply (int *port*, const unsigned char *addr[2]*)

Definition at line 459 of file doya.c.

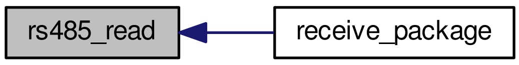
Here is the call graph for this function:



7.33.3.11 static int rs485_read (int *port*, void ∗ *buffer*, int *len*) `[inline]`, `[static]`

Definition at line 135 of file doya.c.

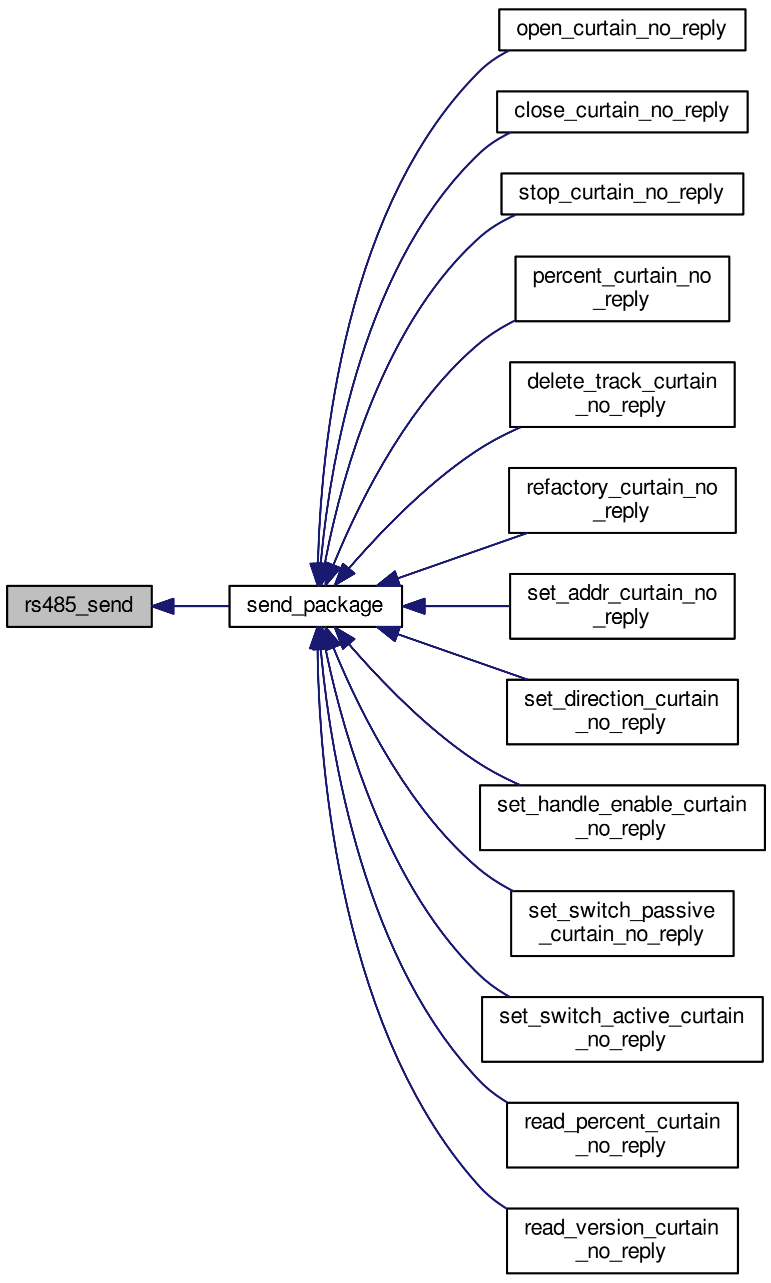
Here is the caller graph for this function:



7.33.3.12 static int rs485_send (int *port,* void ∗ *buffer,* int *len*) [inline], [static]

Definition at line 130 of file doya.c.

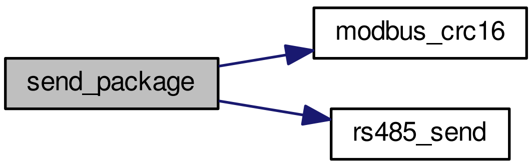
Here is the caller graph for this function:


rs485_send
send_package
open_curtain_no_reply
close_curtain_no_reply
stop_curtain_no_reply
percent_curtain_no_reply
delete_track_curtain_no_reply
refactory_curtain_no_reply
set_addr_curtain_no_reply
set_direction_curtain_no_reply
set_handle_enable_curtain_no_reply
set_switch_passive_curtain_no_reply
set_switch_active_curtain_no_reply
read_percent_curtain_no_reply
read_version_curtain_no_reply

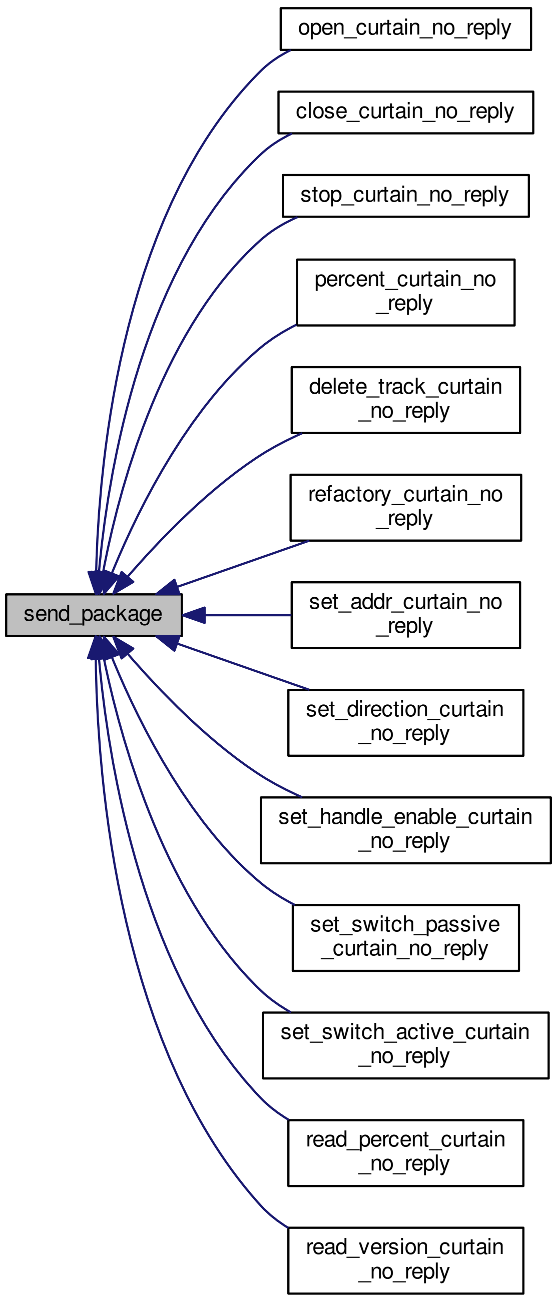
7.33.3.13 static int send_package (int *port*, const struct **package** ∗ *package*) [static]

Definition at line 228 of file doya.c.

Here is the call graph for this function:



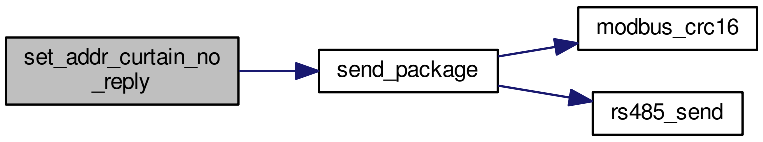
Here is the caller graph for this function:


send_package
open_curtain_no_reply
close_curtain_no_reply
stop_curtain_no_reply
percent_curtain_no _reply
delete_track_curtain _no_reply
refactory_curtain_no _reply
set_addr_curtain_no _reply
set_direction_curtain _no_reply
set_handle_enable_curtain _no_reply
set_switch_passive _curtain_no_reply
set_switch_active_curtain _no_reply
read_percent_curtain _no_reply
read_version_curtain _no_reply

7.33.3.14 int set_addr_curtain_no_reply (int *port,* const unsigned char *addr[2],* unsigned char *set[2]*)

Definition at line 476 of file doya.c.

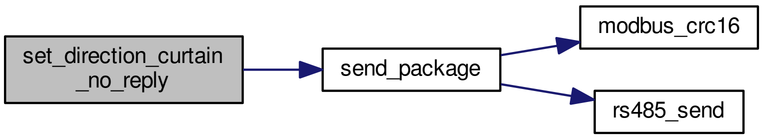
Here is the call graph for this function:



7.33.3.15 int set_direction_curtain_no_reply (int *port,* const unsigned char *addr[2],* unsigned char *same*)

Definition at line 497 of file doya.c.

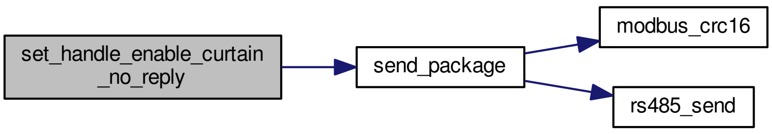
Here is the call graph for this function:



7.33.3.16 int set_handle_enable_curtain_no_reply (int *port,* const unsigned char *addr[2],* unsigned char *enable*)

Definition at line 516 of file doya.c.

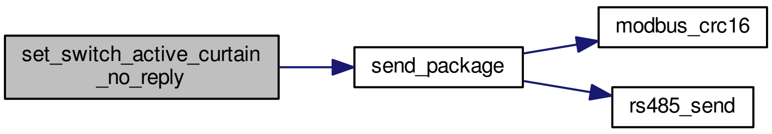
Here is the call graph for this function:



7.33.3.17 int set_switch_active_curtain_no_reply (int *port,* const unsigned char *addr[2],* unsigned char *type*)

Definition at line 554 of file doya.c.

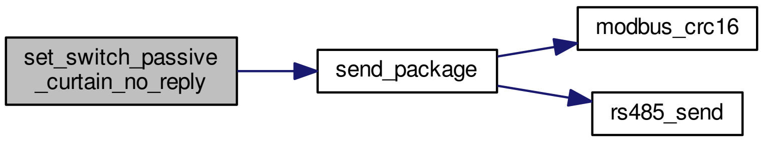
Here is the call graph for this function:



7.33.3.18 int set_switch_passive_curtain_no_reply (int *port*, const unsigned char *addr[2]*, unsigned char *type*)

Definition at line 535 of file doya.c.

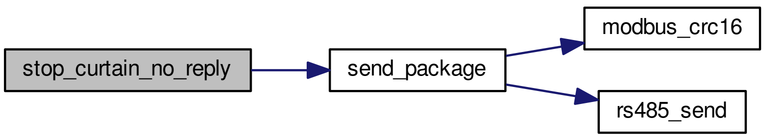
Here is the call graph for this function:



7.33.3.19 int stop_curtain_no_reply (int *port*, const unsigned char *addr[2]*)

Definition at line 406 of file doya.c.

Here is the call graph for this function:



7.33.4 Variable Documentation

7.33.4.1 const unsigned char crc_high[ ] [static]

Initial value:

```
= {
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x00, 0xC1, 0x81, 0x40,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40, 0x01, 0xC0, 0x80, 0x41, 0x01, 0xC0, 0x80, 0x41,
    0x00, 0xC1, 0x81, 0x40
}
```

Definition at line 141 of file doya.c.

7.33.4.2 const unsigned char crc_low[ ] `[static]`

Initial value:

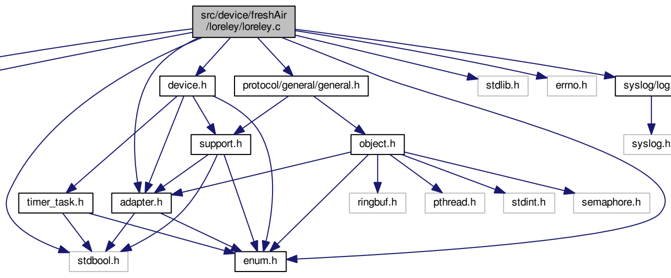
```
= {
    0x00, 0xC0, 0xC1, 0x01, 0xC3, 0x03, 0x02, 0xC2, 0xC6, 0x06, 0x07, 0xC7,
    0x05, 0xC5, 0xC4, 0x04, 0xCC, 0x0C, 0x0D, 0xCD, 0x0F, 0xCF, 0xCE, 0x0E,
    0x0A, 0xCA, 0xCB, 0x0B, 0xC9, 0x09, 0x08, 0xC8, 0xD8, 0x18, 0x19, 0xD9,
    0x1B, 0xDB, 0xDA, 0x1A, 0x1E, 0xDE, 0xDF, 0x1F, 0xDD, 0x1D, 0x1C, 0xDC,
    0x14, 0xD4, 0xD5, 0x15, 0xD7, 0x17, 0x16, 0xD6, 0xD2, 0x12, 0x13, 0xD3,
    0x11, 0xD1, 0xD0, 0x10, 0xF0, 0x30, 0x31, 0xF1, 0x33, 0xF3, 0xF2, 0x32,
    0x36, 0xF6, 0xF7, 0x37, 0xF5, 0x35, 0x34, 0xF4, 0x3C, 0xFC, 0xFD, 0x3D,
    0xFF, 0x3F, 0x3E, 0xFE, 0xFA, 0x3A, 0x3B, 0xFB, 0x39, 0xF9, 0xF8, 0x38,
    0x28, 0xE8, 0xE9, 0x29, 0xEB, 0x2B, 0x2A, 0xEA, 0xEE, 0x2E, 0x2F, 0xEF,
    0x2D, 0xED, 0xEC, 0x2C, 0xE4, 0x24, 0x25, 0xE5, 0x27, 0xE7, 0xE6, 0x26,
    0x22, 0xE2, 0xE3, 0x23, 0xE1, 0x21, 0x20, 0xE0, 0xA0, 0x60, 0x61, 0xA1,
    0x63, 0xA3, 0xA2, 0x62, 0x66, 0xA6, 0xA7, 0x67, 0xA5, 0x65, 0x64, 0xA4,
    0x6C, 0xAC, 0xAD, 0x6D, 0xAF, 0x6F, 0x6E, 0xAE, 0xAA, 0x6A, 0x6B, 0xAB,
    0x69, 0xA9, 0xA8, 0x68, 0x78, 0xB8, 0xB9, 0x79, 0xBB, 0x7B, 0x7A, 0xBA,
    0xBE, 0x7E, 0x7F, 0xBF, 0x7D, 0xBD, 0xBC, 0x7C, 0xB4, 0x74, 0x75, 0xB5,
    0x77, 0xB7, 0xB6, 0x76, 0x72, 0xB2, 0xB3, 0x73, 0xB1, 0x71, 0x70, 0xB0,
    0x50, 0x90, 0x91, 0x51, 0x93, 0x53, 0x52, 0x92, 0x96, 0x56, 0x57, 0x97,
    0x55, 0x95, 0x94, 0x54, 0x9C, 0x5C, 0x5D, 0x9D, 0x5F, 0x9F, 0x9E, 0x5E,
    0x5A, 0x9A, 0x9B, 0x5B, 0x99, 0x59, 0x58, 0x98, 0x88, 0x48, 0x49, 0x89,
    0x4B, 0x8B, 0x8A, 0x4A, 0x4E, 0x8E, 0x8F, 0x4F, 0x8D, 0x4D, 0x4C, 0x8C,
    0x44, 0x84, 0x85, 0x45, 0x87, 0x47, 0x46, 0x86, 0x82, 0x42, 0x43, 0x83,
    0x41, 0x81, 0x80, 0x40
}
```

Definition at line 166 of file doya.c.

7.34 src/device/freshAir/loreley/loreley.c File Reference

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>
#include <errno.h>
#include "enum.h"
#include "syslog/log.h"
#include "adapter.h"
#include "device.h"
#include "protocol/general/general.h"
```

Include dependency graph for loreley.c:



Macros

- #define SOI_SEND 0xaa
- #define SOI_RECEIVE 0x55
- #define ADR_BROADCAST 0xff
- #define ADR_DEFAULT 0x01
- #define RTN_SEND 0x60
- #define RTN_RECEIVE_CMD_RIGHT 0x01
- #define RTN_RECEIVE_CHK_ERROR 0x02
- #define RTN_RECEIVE_CMD_INVALID 0x03
- #define EOI 0x0d
- #define RS485_NEW_TREND_SET_ADDR_CID 0x20
- #define RS485_NEW_TREND_GET_ADDR_CID 0x21
- #define RS485_NEW_TREND_GET_INFO_CID 0x22
- #define RS485_NEW_TREND_SET_ARG_CID 0x23
- #define RS485_NEW_TREND_RESTART_CID 0x24
- #define RS485_NEW_TREND_MODE_WAITING 0x00
- #define RS485_NEW_TREND_MODE_AUTOING 0x01
- #define RS485_NEW_TREND_MODE_OUT_CRC 0x02
- #define RS485_NEW_TREND_MODE_IN_CRC 0x03
- #define RS485_NEW_TREND_MODE_KILLING 0x04
- #define RS485_NEW_TREND_RUN_STATUS_OFF 0x00
- #define RS485_NEW_TREND_RUN_STATUS_ON 0x01
- #define RS485_NEW_TREND_STATUS_ERROR 0x01
- #define RS485_NEW_TREND_STATUS_NORMAL 0x00
- #define NEW_TREND_PACKAGE_MAX 20

Functions

- static unsigned char calculate_sum_check (unsigned char ∗value, int length)
- static int loreley_send_package (unsigned char ∗send_buffer, int send_buffer_len, unsigned char addr, unsigned char msg_cid, const unsigned char ∗data)
- int loreley_send_package_handle (volatile void ∗arg)

  loreley_send_package_handle loreley fresh air package send a buffer
- int loreley_recv_package_handle (volatile void ∗arg)

  loreley_recv_package_handle loreley fresh air process the receive data.

Variables

- static const unsigned char rs485_set_new_trend_table [2][5]

7.34.1 Macro Definition Documentation

7.34.1.1 #define ADR_BROADCAST 0xff

Definition at line 36 of file loreley.c.

7.34.1.2 #define ADR_DEFAULT 0x01

Definition at line 37 of file loreley.c.

7.34.1.3 #define EOI 0x0d

Definition at line 42 of file loreley.c.

7.34.1.4 #define NEW_TREND_PACKAGE_MAX 20

Definition at line 66 of file loreley.c.

7.34.1.5 #define RS485_NEW_TREND_GET_ADDR_CID 0x21

Definition at line 46 of file loreley.c.

7.34.1.6 #define RS485_NEW_TREND_GET_INFO_CID 0x22

Definition at line 47 of file loreley.c.

7.34.1.7 #define RS485_NEW_TREND_MODE_AUTOING 0x01

Definition at line 53 of file loreley.c.

7.34.1.8 #define RS485_NEW_TREND_MODE_IN_CRC 0x03

Definition at line 55 of file loreley.c.

7.34.1.9 #define RS485_NEW_TREND_MODE_KILLING 0x04

Definition at line 56 of file loreley.c.

7.34.1.10 #define RS485_NEW_TREND_MODE_OUT_CRC 0x02

Definition at line 54 of file loreley.c.

7.34.1.11 #define RS485_NEW_TREND_MODE_WAITING 0x00

Definition at line 52 of file loreley.c.

7.34.1.12 #define RS485_NEW_TREND_RESTART_CID 0x24

Definition at line 49 of file loreley.c.

7.34.1.13 #define RS485_NEW_TREND_RUN_STATUS_OFF 0x00

Definition at line 58 of file loreley.c.

7.34.1.14 #define RS485_NEW_TREND_RUN_STATUS_ON 0x01

Definition at line 59 of file loreley.c.

7.34.1.15 #define RS485_NEW_TREND_SET_ADDR_CID 0x20

Definition at line 45 of file loreley.c.

7.34.1.16 #define RS485_NEW_TREND_SET_ARG_CID 0x23

Definition at line 48 of file loreley.c.

7.34.1.17 #define RS485_NEW_TREND_STATUS_ERROR 0x01

Definition at line 61 of file loreley.c.

7.34.1.18 #define RS485_NEW_TREND_STATUS_NORMAL 0x00

Definition at line 62 of file loreley.c.

7.34.1.19 #define RTN_RECEIVE_CHK_ERROR 0x02

Definition at line 40 of file loreley.c.

7.34.1.20 #define RTN_RECEIVE_CMD_INVALID 0x03

Definition at line 41 of file loreley.c.

7.34.1.21 #define RTN_RECEIVE_CMD_RIGHT 0x01

Definition at line 39 of file loreley.c.

7.34.1.22 #define RTN_SEND 0x60

Definition at line 38 of file loreley.c.

7.34.1.23 #define SOI_RECEIVE 0x55

Definition at line 35 of file loreley.c.

7.34.1.24 #define SOI_SEND 0xaa

package protocol

Definition at line 34 of file loreley.c.

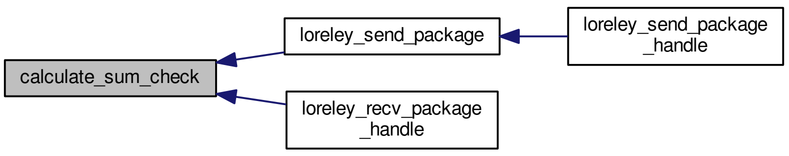
7.34.2 Function Documentation

7.34.2.1 static unsigned char calculate_sum_check (unsigned char ∗ *value,* int *length*) [static]

static function define

Definition at line 77 of file loreley.c.

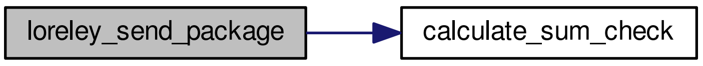
Here is the caller graph for this function:



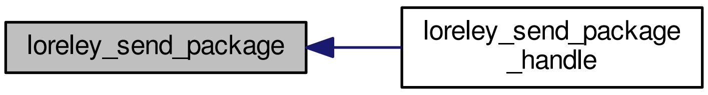
7.34.2.2 static int loreley_send_package (unsigned char ∗ *send_buffer,* int *send_buffer_len,* unsigned char *addr,* unsigned char *msg_cid,* const unsigned char ∗ *data*) [static]

Definition at line 425 of file loreley.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.34.3 Variable Documentation

7.34.3.1 const unsigned char rs485_set_new_trend_table[2][5] [static]

Initial value:

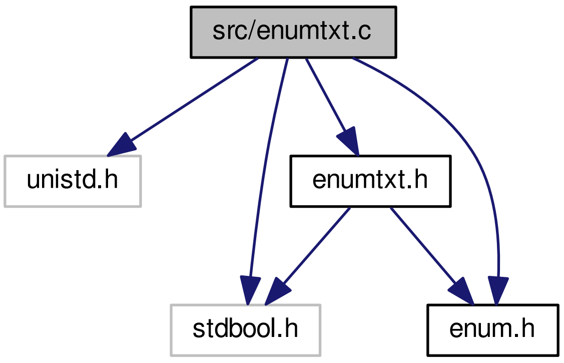
```
= {
{0x01, 0x01, 0x00, 0x00, 0x00},
{0x01, 0x00, 0x00, 0x00, 0x00}
}
```

Definition at line 69 of file loreley.c.

7.35 src/enumtxt.c File Reference

```
#include <unistd.h>
#include <stdbool.h>
#include "enumtxt.h"
#include "enum.h"
```

Include dependency graph for enumtxt.c:



Functions

- char ∗ get_enum_txt_service (rs485_service_type_enum type)

  get_enum_txt_service get enum rs485 service message type
- char ∗ get_enum_txt_rs485_device_type (rs485_device_type_enum type)

  get_enum_txt_rs485_device_type get enum rs485 device type
- char ∗ get_enum_txt_rs485_protocol_type (rs485_protocol_type_enum type)

  get_enum_txt_rs485_protocol_type get enum rs485 protocol type
- char ∗ get_enum_txt_device_method (rs485_device_method_enum type)

  get_enum_txt_device_method get enum device method(command)
- char ∗ get_enum_txt_device_factory (rs485_factory_name_enum name)

  get_enum_txt_device_factory get enum device factory name
- char ∗ get_enum_txt_bool (bool status)

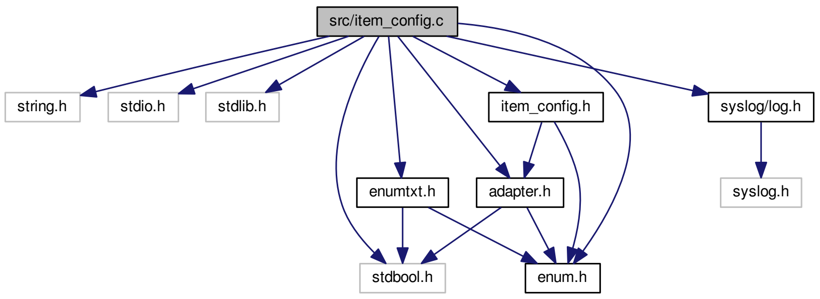
  get_enum_txt_bool get the string about bool value

7.36 src/item_config.c File Reference

```
#include <string.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "item_config.h"
#include "enum.h"
#include "adapter.h"
#include "enumtxt.h"
#include "syslog/log.h"
```

Include dependency graph for item_config.c:



Macros

- #define PANNO_S_ITEM_DEFAULT (1)
- #define PANNO_S_ITEM_WENRUDE (0)
- #define PANNO_S_ITEM_ARMANI (0)
- #define PANNO_S_ITEM_SHAOCHENGGUOJI (0)

Functions

- static void _mount_device (adapter_t ∗adapter, int object_id, rs485_factory_name_enum device, unsigned char addr)
- void panno_s_item_config (adapter_t ∗adapter, rs485_device_type_enum device_type, unsigned char device_addr)

  panno_s_item_config This function is offter the pannoS item config

7.36.1 Macro Definition Documentation

7.36.1.1 #define PANNO_S_ITEM_ARMANI (0)

Definition at line 42 of file item_config.c.

7.36.1.2 #define PANNO_S_ITEM_DEFAULT (1)

Definition at line 32 of file item_config.c.

7.36.1.3 #define PANNO_S_ITEM_SHAOCHENGGUOJI (0)

Definition at line 47 of file item_config.c.

7.36.1.4 #define PANNO_S_ITEM_WENRUDE (0)

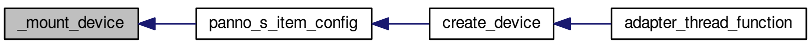
Definition at line 37 of file item_config.c.

7.36.2 Function Documentation

7.36.2.1 static void _mount_device (**adapter_t** ∗ *adapter,* int *object_id,* **rs485_factory_name_enum** *device,* unsigned char *addr*) `[static]`

Definition at line 51 of file item_config.c.

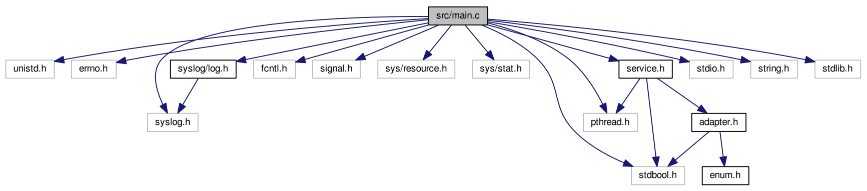
Here is the caller graph for this function:



7.37 src/main.c File Reference

```
#include <unistd.h>
#include <errno.h>
#include <syslog.h>
#include <fcntl.h>
#include <signal.h>
#include <sys/resource.h>
#include <sys/stat.h>
#include <pthread.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include "service.h"
#include "syslog/log.h"
```

Include dependency graph for main.c:



Macros

- #define LOCKFILE "/var/run/rs485.pid"
- #define LOCKMODE (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

Functions

- void ∗ signal_handle_pthread (void ∗arg)
- void daemonize (const char ∗cmd)
- static int lockfile (int fd)
- static int already_running (void)
- int main (int argc, char ∗argv[])

Variables

- static sigset_t mask

7.37.1 Detailed Description

www.enno.com

Date

: Mar 14, 2016

Author

: chuanjiang.wong

Definition in file main.c.

7.37.2 Macro Definition Documentation

7.37.2.1 #define LOCKFILE "/var/run/rs485.pid"

Definition at line 31 of file main.c.

7.37.2.2 #define LOCKMODE (S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)

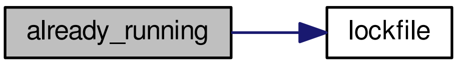
Definition at line 32 of file main.c.

7.37.3 Function Documentation

7.37.3.1 static int already_running (void) [static]

Definition at line 142 of file main.c.

Here is the call graph for this function:



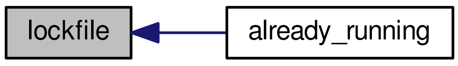
7.37.3.2 void daemonize (const char ∗ *cmd*)

Definition at line 69 of file main.c.

7.37.3.3 static int lockfile (int *fd*) [static]

Definition at line 129 of file main.c.

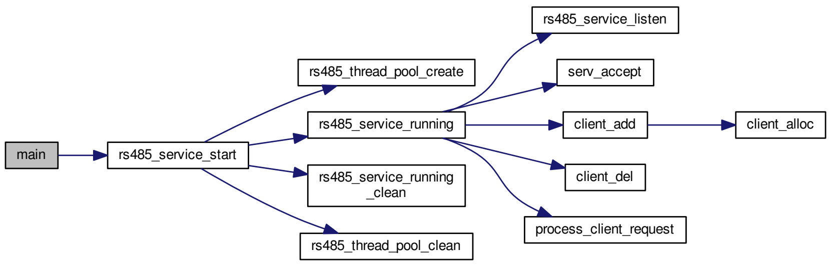
Here is the caller graph for this function:



7.37.3.4 int main (int *argc,* char ∗ *argv[]*)

Definition at line 174 of file main.c.

Here is the call graph for this function:



7.37.3.5 void∗ signal_handle_pthread (void ∗ *arg*)

Definition at line 39 of file main.c.

7.37.4 Variable Documentation

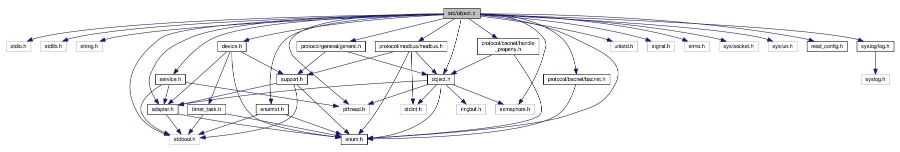
7.37.4.1 sigset_t mask [static]

Definition at line 35 of file main.c.

7.38 src/object.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <pthread.h>
#include <semaphore.h>
#include "read_config.h"
#include "service.h"
#include "syslog/log.h"
#include "adapter.h"
#include "enum.h"
#include "object.h"
#include "protocol/general/general.h"
#include "protocol/bacnet/bacnet.h"
#include "protocol/bacnet/handle_property.h"
#include "protocol/modbus/modbus.h"
#include "device.h"
#include "enumtxt.h"
```

Include dependency graph for object.c:



Macros

- #define RS485_OBJECT_MAX_NUMBERS (5)
- #define RS485_WORK_QUEUE_DEPTH (256)

Functions

- bool check_object_id (int object_id)

    check_object_id check the object is legal
- static int find_available_object_id (void)

    find_available_object_id Find a available object id from object table
- static bool object_is_used (const adapter_t ∗adapter)

    object_is_used To determine whether the object id has been used
- static int work_thread_create (object_management_t ∗object)

    work_thread_create create work thread
- static void work_thread_clean (object_management_t ∗object)

    work_thread_clean clean the work thread
- static bool check_object_is_support (rs485_protocol_type_enum object_type)
- int create_object (const adapter_t ∗adapter)

    create_object create a object by the adapter message

- int delete_object (int object_id)

    delete_object delete a rs485 object by object id
- int get_object_type (int object_id)

    get_object_type get the object protocol type
- int get_object_mount_device (int object_id, int ∗out_id, int out_id_len)

    get_object_mount_device get the object mount device
- int object_mount_device_id (int object_id, int device_id)

    object_mount_device_id add a device to his object
- void object_unmount_device_id (int object_id, int device_id)

    object_unmount_device_id delete a device form his object
- bool check_object_numbers_have_idle (int object_id)

    check_object_numbers_have_idle check object mount device is full ?
- void ∗ get_object_work_queue (int object_id)

    get_object_work_queue get the object of work queue
- void ∗ get_object_queue_sem (int object_id)

    get_object_queue_sem get the object of work queue semphore

Variables

- static object_management_t ∗ glb_object_manage [RS485_OBJECT_MAX_NUMBERS] = { 0 }
- static struct ring_buffer_t glb_work_queue [RS485_OBJECT_MAX_NUMBERS]

7.38.1 Macro Definition Documentation

7.38.1.1 #define RS485_OBJECT_MAX_NUMBERS (5)

Definition at line 47 of file object.c.

7.38.1.2 #define RS485_WORK_QUEUE_DEPTH (256)

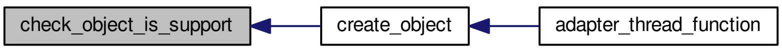
Definition at line 51 of file object.c.

7.38.2 Function Documentation

7.38.2.1 static bool check_object_is_support (rs485_protocol_type_enum *object_type*) [static]

Definition at line 311 of file object.c.

Here is the caller graph for this function:



7.38.3 Variable Documentation

7.38.3.1 object_management_t∗ glb_object_manage[RS485_OBJECT_MAX_NUMBERS] = { 0 } `[static]`

define the RS485 object management table

Definition at line 56 of file object.c.

7.38.3.2 struct ring_buffer_t glb_work_queue[RS485_OBJECT_MAX_NUMBERS] `[static]`

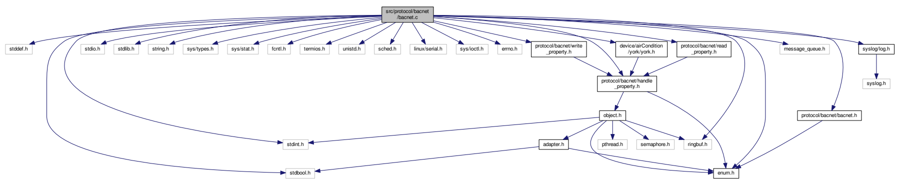
define the RS485 object ring buffer

Definition at line 59 of file object.c.

7.39 src/protocol/bacnet/bacnet.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <sched.h>
#include <linux/serial.h>
#include <sys/ioctl.h>
#include <errno.h>
#include "enum.h"
#include "protocol/bacnet/bacnet.h"
#include "protocol/bacnet/handle_property.h"
#include "protocol/bacnet/read_property.h"
#include "protocol/bacnet/write_property.h"
#include "message_queue.h"
#include "syslog/log.h"
#include "ringbuf.h"
#include "device/airCondition/york/york.h"
```

Include dependency graph for bacnet.c:



Functions

- static int york_air_condition_handle (const struct bacnet ∗handle)
- void ∗ bacnet_work_thread_function (void ∗arg)

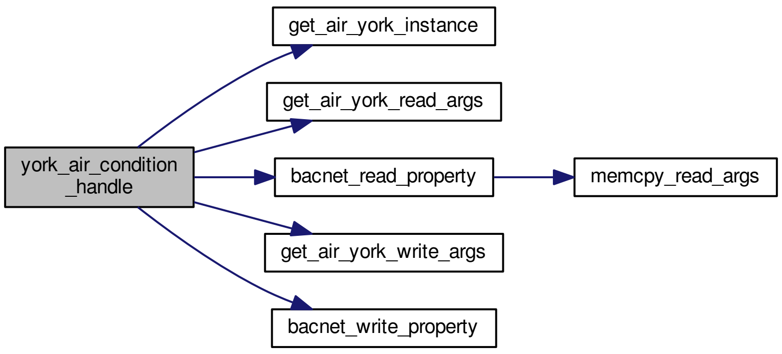
  bacnet_work_thread_function The bacnet work thread

7.39.1 Function Documentation

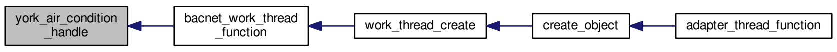
7.39.1.1 static int york_air_condition_handle (const struct **bacnet** * *handle*) [static]

Definition at line 51 of file bacnet.c.

Here is the call graph for this function:



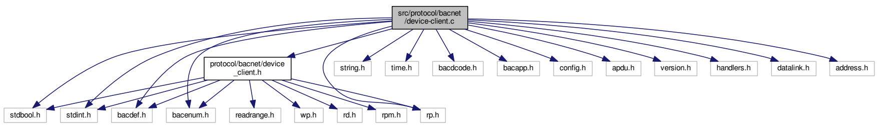
Here is the caller graph for this function:



7.40 src/protocol/bacnet/device-client.c File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include <string.h>
#include <time.h>
#include "bacdef.h"
#include "bacdcode.h"
#include "bacenum.h"
#include "bacapp.h"
#include "config.h"
#include "apdu.h"
#include "rp.h"
#include "version.h"
#include "handlers.h"
#include "datalink.h"
#include "address.h"
#include "protocol/bacnet/device_client.h"
```

Include dependency graph for device-client.c:



Functions

- int Device_Read_Property_Local (BACNET_READ_PROPERTY_DATA ∗rpdata)
- static struct object_functions ∗ Device_Objects_Find_Functions (BACNET_OBJECT_TYPE Object_Type)
- unsigned Device_Count (void)
- uint32_t Device_Index_To_Instance (unsigned index)
- uint32_t Device_Object_Instance_Number (void)
- bool Device_Set_Object_Instance_Number (uint32_t object_id)
- bool Device_Valid_Object_Instance_Number (uint32_t object_id)
- bool Device_Object_Name (uint32_t object_instance, BACNET_CHARACTER_STRING ∗object_name)
- bool Device_Set_Object_Name (BACNET_CHARACTER_STRING ∗object_name)
- BACNET_DEVICE_STATUS Device_System_Status (void)
- int Device_Set_System_Status (BACNET_DEVICE_STATUS status, bool local)
- const char ∗ Device_Vendor_Name (void)
- uint16_t Device_Vendor_Identifier (void)
- void Device_Set_Vendor_Identifier (uint16_t vendor_id)
- const char ∗ Device_Model_Name (void)
- bool Device_Set_Model_Name (const char ∗name, size_t length)
- const char ∗ Device_Firmware_Revision (void)
- const char ∗ Device_Application_Software_Version (void)
- bool Device_Set_Application_Software_Version (const char ∗name, size_t length)
- const char ∗ Device_Description (void)
- bool Device_Set_Description (const char ∗name, size_t length)
- const char ∗ Device_Location (void)
- bool Device_Set_Location (const char ∗name, size_t length)
- uint8_t Device_Protocol_Version (void)
- uint8_t Device_Protocol_Revision (void)
- BACNET_SEGMENTATION Device_Segmentation_Supported (void)
- uint32_t Device_Database_Revision (void)
- void Device_Set_Database_Revision (uint32_t revision)
- void Device_Inc_Database_Revision (void)
- unsigned Device_Object_List_Count (void)
- bool Device_Object_List_Identifier (unsigned array_index, int ∗object_type, uint32_t ∗instance)
- bool Device_Valid_Object_Name (BACNET_CHARACTER_STRING ∗object_name1, int ∗object_type, uint32_t ∗object_instance)
- bool Device_Valid_Object_Id (int object_type, uint32_t object_instance)
- bool Device_Object_Name_Copy (BACNET_OBJECT_TYPE object_type, uint32_t object_instance, BACN←ET_CHARACTER_STRING ∗object_name)
- int Device_Read_Property (BACNET_READ_PROPERTY_DATA ∗rpdata)
- void Device_Init (object_functions_t ∗object_table)

Variables

- static uint32_t Object_Instance_Number = 260001
- static BACNET_CHARACTER_STRING My_Object_Name
- static BACNET_DEVICE_STATUS System_Status = STATUS_OPERATIONAL
- static char ∗ Vendor_Name = BACNET_VENDOR_NAME
- static uint16_t Vendor_Identifier = BACNET_VENDOR_ID
- static char ∗ Model_Name = "GNU"
- static char ∗ Application_Software_Version = "1.0"
- static char ∗ Location = "USA"
- static char ∗ Description = "command line client"
- static uint32_t Database_Revision = 0
- static object_functions_t Object_Table [ ]

7.40.1 Detailed Description

Lightweight base "class" for handling all BACnet objects belonging to a BACnet device, as well as Device-specific properties. This Device instance is designed to meet minimal functionality for simple clients.

Definition in file device-client.c.

7.40.2 Function Documentation

7.40.2.1 const char∗ Device_Application_Software_Version (void)

Definition at line 362 of file device-client.c.

7.40.2.2 unsigned Device_Count (void)

Definition at line 163 of file device-client.c.

7.40.2.3 uint32_t Device_Database_Revision (void)

Definition at line 443 of file device-client.c.

7.40.2.4 const char∗ Device_Description (void)

Definition at line 383 of file device-client.c.

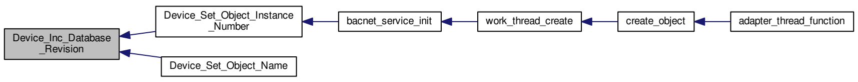
7.40.2.5 const char∗ Device_Firmware_Revision (void)

Definition at line 356 of file device-client.c.

7.40.2.6 void Device_Inc_Database_Revision (void)

Definition at line 460 of file device-client.c.

Here is the caller graph for this function:



7.40.2.7 uint32_t Device_Index_To_Instance (unsigned *index*)

Definition at line 169 of file device-client.c.

7.40.2.8 void Device_Init (object_functions_t ∗ *object_table*)

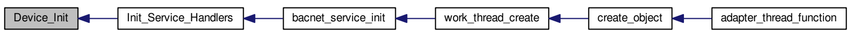
Initialize the Device Object. Initialize the group of object helper functions for any supported Object. Initialize each of the Device Object child Object instances.

Parameters

object_table	[in,out] array of structure with object functions. Each Child Object must provide some implementation of each of these functions in order to properly support the default handlers.

Definition at line 895 of file device-client.c.

Here is the caller graph for this function:



7.40.2.9 const char∗ Device_Location (void)

Definition at line 404 of file device-client.c.

7.40.2.10 const char∗ Device_Model_Name (void)

Definition at line 335 of file device-client.c.

7.40.2.11 uint32_t Device_Object_Instance_Number (void)

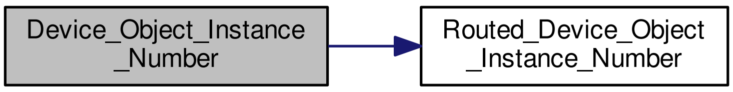
Return the Object Instance number for our (single) Device Object. This is a key function, widely invoked by the handler code, since it provides "our" (ie, local) address.

Returns

The Instance number used in the BACNET_OBJECT_ID for the Device.

Definition at line 184 of file device-client.c.

Here is the call graph for this function:



7.40.2.12 unsigned Device_Object_List_Count (void)

Get the total count of objects supported by this Device Object.

Note

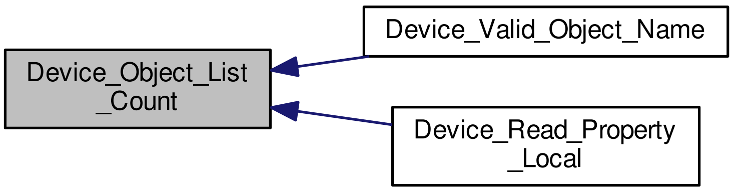
Since many network clients depend on the object list for discovery, it must be consistent!

Returns

The count of objects, for all supported Object types.

Definition at line 471 of file device-client.c.

Here is the caller graph for this function:



7.40.2.13 bool Device_Object_List_Identifier (unsigned *array_index,* int ∗ *object_type,* uint32_t ∗ *instance*)

Lookup the Object at the given array index in the Device's Object List. Even though we don't keep a single linear array of objects in the Device, this method acts as though we do and works through a virtual, concatenated array of all of our object type arrays.

Parameters

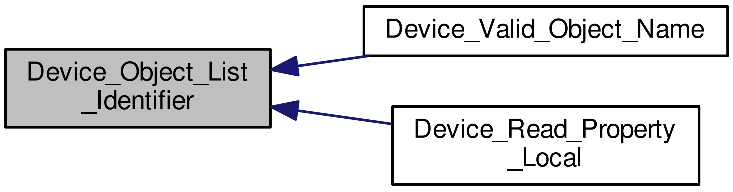
array_index	[in] The desired array index (1 to N)
object_type	[out] The object's type, if found.
instance	[out] The object's instance number, if found.

Returns

True if found, else false.

Definition at line 499 of file device-client.c.

Here is the caller graph for this function:



7.40.2.14 bool Device_Object_Name (uint32_t *object_instance,* BACNET_CHARACTER_STRING ∗ *object_name*)

Definition at line 215 of file device-client.c.

7.40.2.15 bool Device_Object_Name_Copy (BACNET_OBJECT_TYPE *object_type,* uint32_t *object_instance,* BACNET_CHARACTER_STRING ∗ *object_name*)

Copy a child object's object_name value, given its ID.

Parameters

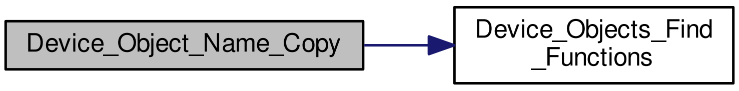
object_type	[in] The BACNET_OBJECT_TYPE of the child Object.
object_instance	[in] The object instance number of the child Object.
object_name	[out] The Object Name found for this child Object.

Returns

True on success or else False if not found.

Definition at line 620 of file device-client.c.

Here is the call graph for this function:



7.40.2.16 static struct **object_functions**∗ Device_Objects_Find_Functions (BACNET_OBJECT_TYPE *Object_Type*) `[static]`

Glue function to let the Device object, when called by a handler, lookup which Object type needs to be invoked.

Parameters

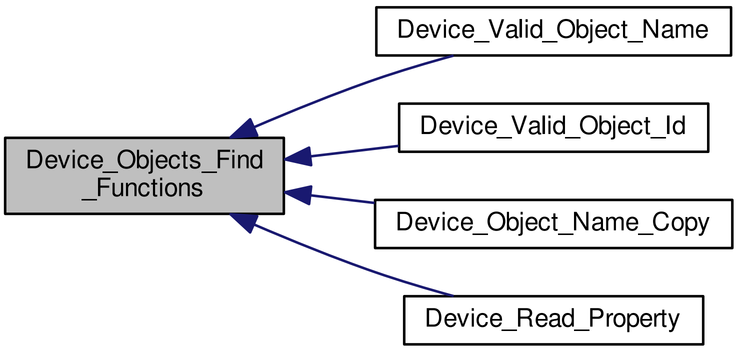
Object_Type	[in] The type of BACnet Object the handler wants to access.

Returns

Pointer to the group of object helper functions that implement this type of Object.

Definition at line 145 of file device-client.c.

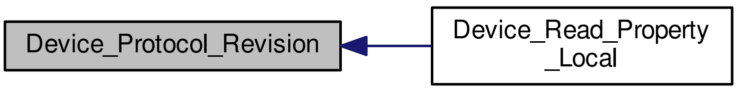
Here is the caller graph for this function:



7.40.2.17 uint8_t Device_Protocol_Revision (void)

Definition at line 431 of file device-client.c.

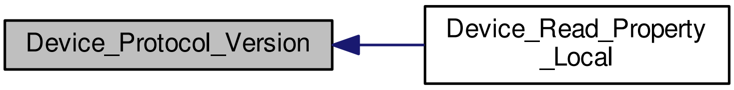
Here is the caller graph for this function:



7.40.2.18 uint8_t Device_Protocol_Version (void)

Definition at line 425 of file device-client.c.

Here is the caller graph for this function:



7.40.2.19 int Device_Read_Property (BACNET_READ_PROPERTY_DATA ∗ *rpdata*)

Looks up the requested Object and Property, and encodes its Value in an APDU.

If the Object or Property can't be found, sets the error class and code.

Parameters

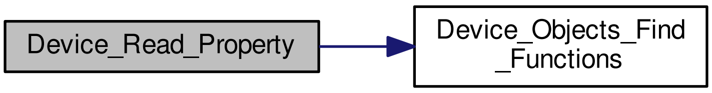
rpdata	[in,out] Structure with the desired Object and Property info on entry, and APDU message on return.

Returns

The length of the APDU on success, else BACNET_STATUS_ERROR

Definition at line 859 of file device-client.c.

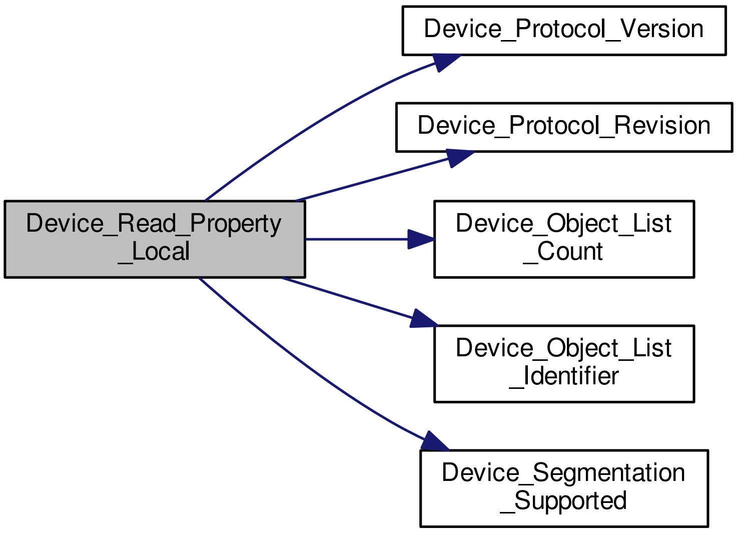
Here is the call graph for this function:



7.40.2.20 int Device_Read_Property_Local (BACNET_READ_PROPERTY_DATA ∗ *rpdata*)

Definition at line 638 of file device-client.c.

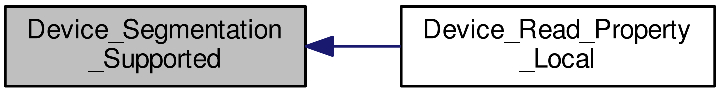
Here is the call graph for this function:



7.40.2.21 BACNET_SEGMENTATION Device_Segmentation_Supported (void)

Definition at line 437 of file device-client.c.

Here is the caller graph for this function:



7.40.2.22 bool Device_Set_Application_Software_Version (const char ∗ *name,* size_t *length*)

Definition at line 368 of file device-client.c.

7.40.2.23 void Device_Set_Database_Revision (uint32_t *revision*)

Definition at line 449 of file device-client.c.

7.40.2.24 bool Device_Set_Description (const char ∗ *name,* size_t *length*)

Definition at line 389 of file device-client.c.

7.40.2.25 bool Device_Set_Location (const char ∗ *name,* size_t *length*)

Definition at line 410 of file device-client.c.

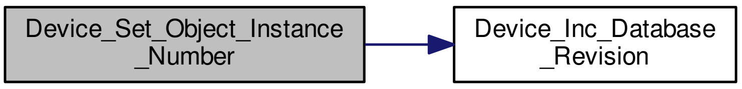
7.40.2.26 bool Device_Set_Model_Name (const char ∗ *name,* size_t *length*)

Definition at line 341 of file device-client.c.

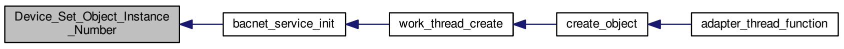
7.40.2.27 bool Device_Set_Object_Instance_Number (uint32_t *object_id*)

Definition at line 194 of file device-client.c.

Here is the call graph for this function:


Device_Set_Object_Instance_Number
Device_Inc_Database_Revision

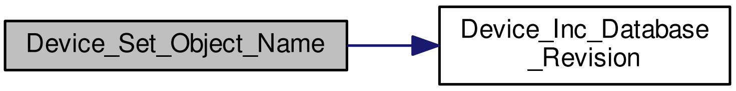
Here is the caller graph for this function:



7.40.2.28 bool Device_Set_Object_Name (BACNET_CHARACTER_STRING ∗ *object_name*)

Definition at line 228 of file device-client.c.

Here is the call graph for this function:



7.40.2.29 int Device_Set_System_Status (BACNET_DEVICE_STATUS *status*, bool *local*)

Definition at line 248 of file device-client.c.

7.40.2.30 void Device_Set_Vendor_Identifier (uint16_t *vendor_id*)

Definition at line 329 of file device-client.c.

7.40.2.31 BACNET_DEVICE_STATUS Device_System_Status (void)

Definition at line 242 of file device-client.c.

7.40.2.32 bool Device_Valid_Object_Id (int *object_type*, uint32_t *object_instance*)

Determine if we have an object of this type and instance number.

Parameters

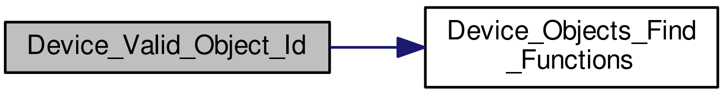
object_type	[in] The desired BACNET_OBJECT_TYPE
object_instance	[in] The object instance number to be looked up.

Returns

True if found, else False if no such Object in this device.

Definition at line 599 of file device-client.c.

Here is the call graph for this function:



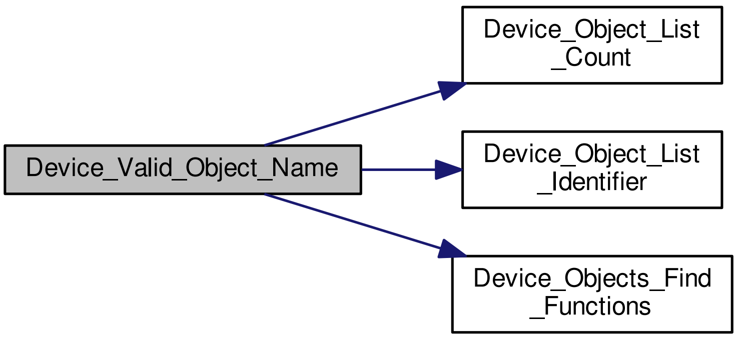
7.40.2.33 bool Device_Valid_Object_Instance_Number (uint32_t *object_id*)

Definition at line 209 of file device-client.c.

7.40.2.34 bool Device_Valid_Object_Name (BACNET_CHARACTER_STRING ∗ *object_name1*, int ∗ *object_type*, uint32_t ∗ *object_instance*)

Definition at line 558 of file device-client.c.

Here is the call graph for this function:


Device_Valid_Object_Name
Device_Object_List _Count
Device_Object_List _Identifier
Device_Objects_Find _Functions

7.40.2.35 uint16_t Device_Vendor_Identifier (void)

Returns the Vendor ID for this Device. See the assignments at `http://www.bacnet.org/VendorID/BA←Cnet%20Vendor%20IDs.htm`

Returns

The Vendor ID of this Device.

Definition at line 323 of file device-client.c.

7.40.2.36 const char∗ Device_Vendor_Name (void)

Definition at line 313 of file device-client.c.

7.40.3 Variable Documentation

7.40.3.1 char∗ Application_Software_Version = "1.0" `[static]`

Definition at line 60 of file device-client.c.

7.40.3.2 uint32_t Database_Revision = 0 `[static]`

Definition at line 85 of file device-client.c.

7.40.3.3 char∗ Description = "command line client" `[static]`

Definition at line 62 of file device-client.c.

7.40.3.4 char∗ Location = "USA" `[static]`

Definition at line 61 of file device-client.c.

7.40.3.5 char∗ Model_Name = "GNU" [static]

Definition at line 59 of file device-client.c.

7.40.3.6 BACNET_CHARACTER_STRING My_Object_Name [static]

Definition at line 55 of file device-client.c.

7.40.3.7 uint32_t Object_Instance_Number = 260001 [static]

Definition at line 54 of file device-client.c.

7.40.3.8 object_functions_t Object_Table[ ] [static]

Definition at line 105 of file device-client.c.

7.40.3.9 BACNET_DEVICE_STATUS System_Status = STATUS_OPERATIONAL [static]

Definition at line 56 of file device-client.c.

7.40.3.10 uint16_t Vendor_Identifier = BACNET_VENDOR_ID [static]

Definition at line 58 of file device-client.c.

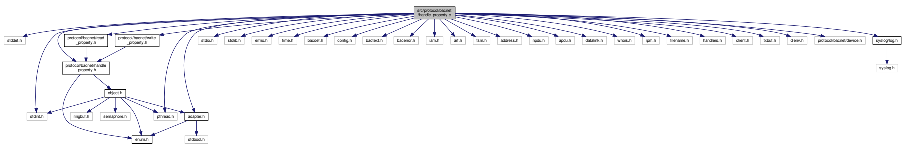
7.40.3.11 char∗ Vendor_Name = BACNET_VENDOR_NAME [static]

Definition at line 57 of file device-client.c.

7.41 src/protocol/bacnet/handle_property.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include "bacdef.h"
#include "config.h"
#include "bactext.h"
#include "bacerror.h"
#include "iam.h"
#include "arf.h"
#include "tsm.h"
#include "address.h"
#include "npdu.h"
#include "apdu.h"
#include "datalink.h"
#include "whois.h"
#include "rpm.h"
#include "filename.h"
#include "handlers.h"
#include "client.h"
#include "txbuf.h"
#include "dlenv.h"
#include <pthread.h>
#include "protocol/bacnet/handle_property.h"
#include "protocol/bacnet/read_property.h"
#include "protocol/bacnet/write_property.h"
#include "protocol/bacnet/device.h"
#include "adapter.h"
#include "syslog/log.h"
```

Include dependency graph for handle_property.c:



Macros

- #define MAX_PROPERTY_VALUES 64

Functions

- void My_Read_Property_Ack_Handler (uint8_t ∗service_request, uint16_t service_len, BACNET_ADDRE↩ SS ∗src, BACNET_CONFIRMED_SERVICE_ACK_DATA ∗service_data)
- void MyWritePropertySimpleAckHandler (BACNET_ADDRESS ∗src, uint8_t invoke_id)
- void My_Read_Property_Multiple_Ack_Handler (uint8_t ∗service_request, uint16_t service_len, BACNET↩ _ADDRESS ∗src, BACNET_CONFIRMED_SERVICE_ACK_DATA ∗service_data)
- static void MyErrorHandler (BACNET_ADDRESS ∗src, uint8_t invoke_id, BACNET_ERROR_CLASS error↩ _class, BACNET_ERROR_CODE error_code)
- static void MyAbortHandler (BACNET_ADDRESS ∗src, uint8_t invoke_id, uint8_t abort_reason, bool server)

- static void MyRejectHandler (BACNET_ADDRESS ∗src, uint8_t invoke_id, uint8_t reject_reason)
- static void Init_Service_Handlers (void)
- static int enno_dlenv_init (const object_management_t ∗object)
- int bacnet_service_init (object_management_t ∗object)

  bacnet_service_init bacnet physics initialze.

Variables

- int glb_config_bacnet_object_instance

7.41.1 Macro Definition Documentation

7.41.1.1 #define MAX_PROPERTY_VALUES 64

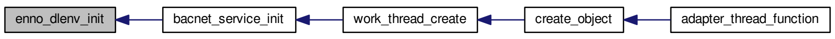
Definition at line 45 of file handle_property.c.

7.41.2 Function Documentation

7.41.2.1 static int enno_dlenv_init (const object_management_t ∗ *object*) [static]

Definition at line 116 of file handle_property.c.

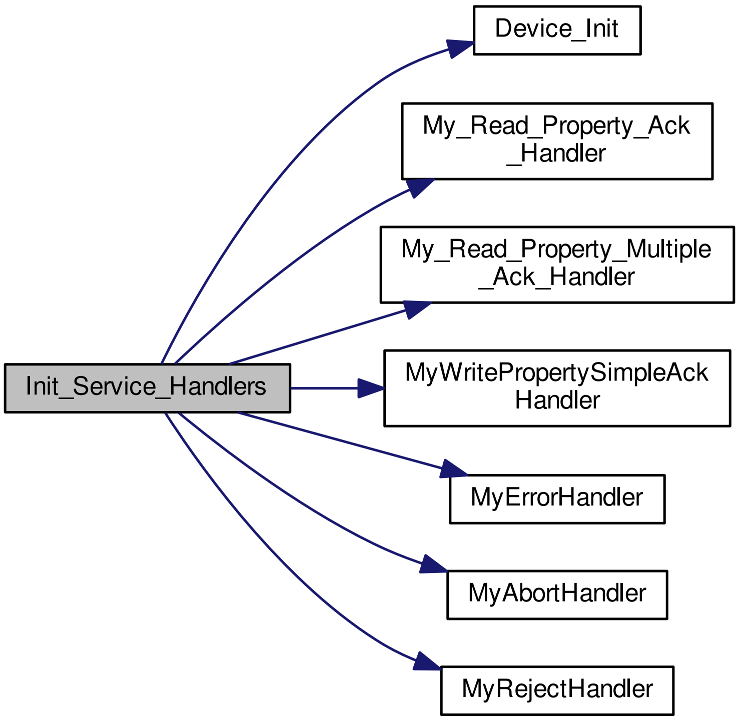
Here is the caller graph for this function:



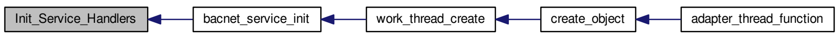
7.41.2.2 static void Init_Service_Handlers (void) [static]

Definition at line 85 of file handle_property.c.

Here is the call graph for this function:


Device_Init
My_Read_Property_Ack _Handler
My_Read_Property_Multiple _Ack_Handler
Init_Service_Handlers
MyWritePropertySimpleAck Handler
MyErrorHandler
MyAbortHandler
MyRejectHandler

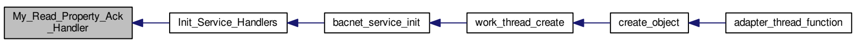
Here is the caller graph for this function:



7.41.2.3 void My_Read_Property_Ack_Handler (uint8_t ∗ *service_request,* uint16_t *service_len,* BACNET_ADDRESS ∗ *src,* BACNET_CONFIRMED_SERVICE_ACK_DATA ∗ *service_data*)

Definition at line 48 of file read_property.c.

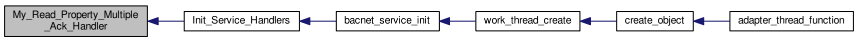
Here is the caller graph for this function:



7.41.2.4 void My_Read_Property_Multiple_Ack_Handler (uint8_t ∗ *service_request,* uint16_t *service_len,* BACNET_ADDRESS ∗ *src,* BACNET_CONFIRMED_SERVICE_ACK_DATA ∗ *service_data*)

Definition at line 82 of file read_property.c.

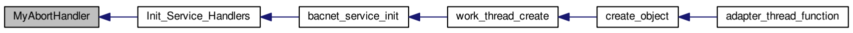
Here is the caller graph for this function:



7.41.2.5 static void MyAbortHandler (BACNET_ADDRESS ∗ *src,* uint8_t *invoke_id,* uint8_t *abort_reason,* bool *server*) [static]

Definition at line 71 of file handle_property.c.

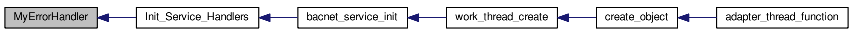
Here is the caller graph for this function:



7.41.2.6 static void MyErrorHandler (BACNET_ADDRESS ∗ *src,* uint8_t *invoke_id,* BACNET_ERROR_CLASS *error_class,* BACNET_ERROR_CODE *error_code*) [static]

Definition at line 65 of file handle_property.c.

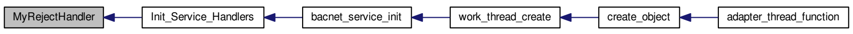
Here is the caller graph for this function:



7.41.2.7 static void MyRejectHandler (BACNET_ADDRESS ∗ *src,* uint8_t *invoke_id,* uint8_t *reject_reason*) [static]

Definition at line 78 of file handle_property.c.

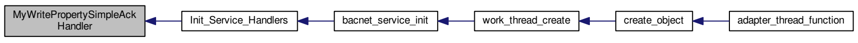
Here is the caller graph for this function:



7.41.2.8 void MyWritePropertySimpleAckHandler (BACNET_ADDRESS ∗ *src,* uint8_t *invoke_id*)

Definition at line 55 of file write_property.c.

Here is the caller graph for this function:



7.41.3 Variable Documentation

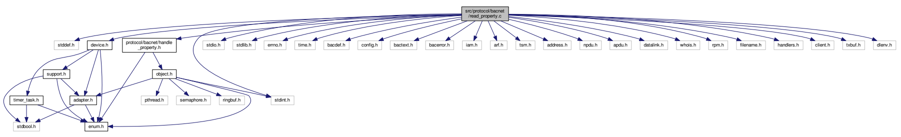
7.41.3.1 int glb_config_bacnet_object_instance

Definition at line 35 of file read_config.c.

7.42 src/protocol/bacnet/read_property.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include "bacdef.h"
#include "config.h"
#include "bactext.h"
#include "bacerror.h"
#include "iam.h"
#include "arf.h"
#include "tsm.h"
#include "address.h"
#include "npdu.h"
#include "apdu.h"
#include "device.h"
#include "datalink.h"
#include "whois.h"
#include "rpm.h"
#include "filename.h"
#include "handlers.h"
#include "client.h"
#include "txbuf.h"
#include "dlenv.h"
#include "protocol/bacnet/handle_property.h"
```

Include dependency graph for read_property.c:



Functions

- void My_Read_Property_Ack_Handler (uint8_t ∗service_request, uint16_t service_len, BACNET_ADDRE←SS ∗src, BACNET_CONFIRMED_SERVICE_ACK_DATA ∗service_data)

- void My_Read_Property_Multiple_Ack_Handler (uint8_t ∗service_request, uint16_t service_len, BACNET↩_ADDRESS ∗src, BACNET_CONFIRMED_SERVICE_ACK_DATA ∗service_data)
- static int memcpy_read_args (bacnet_read_args_t ∗args)
- int bacnet_read_property (bacnet_read_args_t ∗args)

Variables

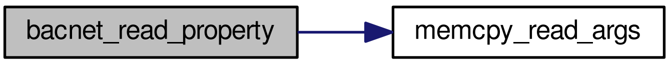
- static BACNET_READ_ACCESS_DATA read_access_data [BACNET_READ_ARGS_OBJECT_MAX]
- static BACNET_PROPERTY_REFERENCE read_access_data_property [BACNET_READ_ARGS_OBJE↩CT_MAX]
- static uint8_t Request_Invoke_ID = 0
- static BACNET_ADDRESS Target_Address

7.42.1 Function Documentation

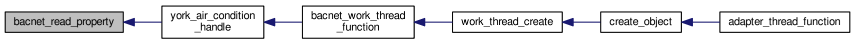
7.42.1.1 int bacnet_read_property (**bacnet_read_args_t** ∗ ***args***)

Definition at line 152 of file read_property.c.

Here is the call graph for this function:



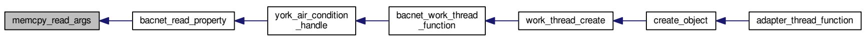
Here is the caller graph for this function:



7.42.1.2 static int memcpy_read_args (**bacnet_read_args_t** ∗ ***args***) [static]

Definition at line 115 of file read_property.c.

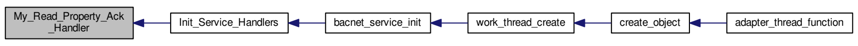
Here is the caller graph for this function:



7.42.1.3 void My_Read_Property_Ack_Handler (uint8_t ∗ *service_request,* uint16_t *service_len,* BACNET_ADDRESS ∗ *src,* BACNET_CONFIRMED_SERVICE_ACK_DATA ∗ *service_data*)

Definition at line 48 of file read_property.c.

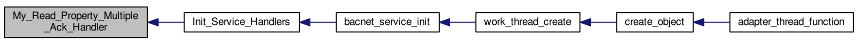
Here is the caller graph for this function:



7.42.1.4 void My_Read_Property_Multiple_Ack_Handler (uint8_t ∗ *service_request,* uint16_t *service_len,* BACNET_ADDRESS ∗ *src,* BACNET_CONFIRMED_SERVICE_ACK_DATA ∗ *service_data*)

Definition at line 82 of file read_property.c.

Here is the caller graph for this function:



7.42.2 Variable Documentation

7.42.2.1 BACNET_READ_ACCESS_DATA read_access_data[BACNET_READ_ARGS_OBJECT_MAX] [static]

Definition at line 39 of file read_property.c.

7.42.2.2 BACNET_PROPERTY_REFERENCE read_access_data_property[BACNET_READ_ARGS_OBJECT_MAX] [static]

Definition at line 40 of file read_property.c.

7.42.2.3 uint8_t Request_Invoke_ID = 0 [static]

Definition at line 44 of file read_property.c.

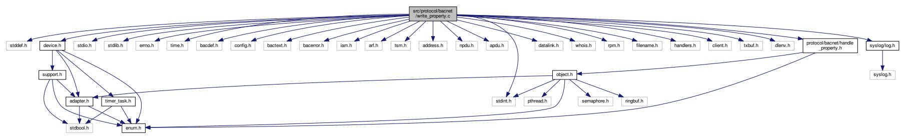
7.42.2.4 BACNET_ADDRESS Target_Address [static]

Definition at line 45 of file read_property.c.

7.43 src/protocol/bacnet/write_property.c File Reference

```
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <time.h>
#include "bacdef.h"
#include "config.h"
#include "bactext.h"
#include "bacerror.h"
#include "iam.h"
#include "arf.h"
#include "tsm.h"
#include "address.h"
#include "npdu.h"
#include "apdu.h"
#include "device.h"
#include "datalink.h"
#include "whois.h"
#include "rpm.h"
#include "filename.h"
#include "handlers.h"
#include "client.h"
#include "txbuf.h"
#include "dlenv.h"
#include "protocol/bacnet/handle_property.h"
#include "syslog/log.h"
```

Include dependency graph for write_property.c:



Macros

- #define MAX_PROPERTY_VALUES 64
- #define RETRANSMISSION_TIMES 3

Functions

- void MyWritePropertySimpleAckHandler (BACNET_ADDRESS ∗src, uint8_t invoke_id)
- int bacnet_write_property (const bacnet_write_args_t ∗args)

Variables

- static uint8_t Rx_Buf [MAX_MPDU] = { 0 }
- static
  BACNET_APPLICATION_DATA_VALUE Target_Object_Property_Value [MAX_PROPERTY_VALUES]
- static uint8_t Request_Invoke_ID = 0
- static BACNET_ADDRESS Target_Address
- static bool Error_Detected = false

7.43.1 Macro Definition Documentation

7.43.1.1 #define MAX_PROPERTY_VALUES 64

Definition at line 40 of file write_property.c.

7.43.1.2 #define RETRANSMISSION_TIMES 3

Definition at line 44 of file write_property.c.

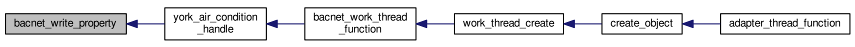
7.43.2 Function Documentation

7.43.2.1 int bacnet_write_property (const bacnet_write_args_t ∗ *args*)

500ms∗4, 2s

Definition at line 63 of file write_property.c.

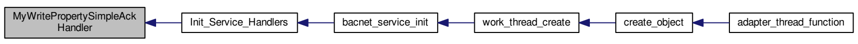
Here is the caller graph for this function:



7.43.2.2 void MyWritePropertySimpleAckHandler (BACNET_ADDRESS ∗ *src,* uint8_t *invoke_id*)

Definition at line 55 of file write_property.c.

Here is the caller graph for this function:



7.43.3 Variable Documentation

7.43.3.1 bool Error_Detected = false [static]

Definition at line 53 of file write_property.c.

7.43.3.2 uint8_t Request_Invoke_ID = 0 [static]

Definition at line 51 of file write_property.c.

7.43.3.3 uint8_t Rx_Buf[MAX_MPDU] = { 0 } [static]

Definition at line 47 of file write_property.c.

7.43.3.4 BACNET_ADDRESS Target_Address `[static]`

Definition at line 52 of file write_property.c.

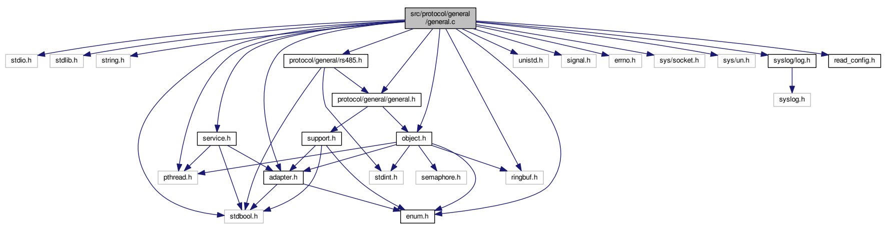
7.43.3.5 BACNET_APPLICATION_DATA_VALUE Target_Object_Property_Value[MAX_PROPERTY_VALUES] `[static]`

Definition at line 49 of file write_property.c.

7.44 src/protocol/general/general.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <pthread.h>
#include "service.h"
#include "syslog/log.h"
#include "adapter.h"
#include "enum.h"
#include "ringbuf.h"
#include "protocol/general/general.h"
#include "protocol/general/rs485.h"
#include "object.h"
#include "read_config.h"
```

Include dependency graph for general.c:



Macros

- #define BUS_MAX_RETRANSMISSION (3)

Functions

- int general_service_init (object_management_t ∗object)

  general_service_init The general protocol(user defined) initilize
- void ∗ general_work_thread_function (void ∗arg)

  general_work_thread_function The general work thread function

7.44.1 Macro Definition Documentation

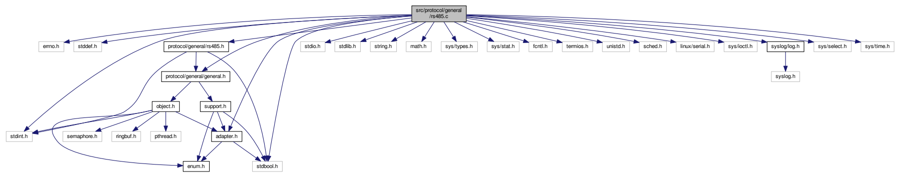
7.44.1.1 #define BUS_MAX_RETRANSMISSION (3)

Definition at line 42 of file general.c.

7.45 src/protocol/general/rs485.c File Reference

```
#include <errno.h>
#include <stddef.h>
#include <stdint.h>
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <unistd.h>
#include <sched.h>
#include <linux/serial.h>
#include <sys/ioctl.h>
#include "adapter.h"
#include "protocol/general/general.h"
#include "protocol/general/rs485.h"
#include "syslog/log.h"
#include <sys/select.h>
#include <sys/time.h>
```

Include dependency graph for rs485.c:



Macros

- #define RS485_DEBUG (1)
- #define RS485MOD 0
- #define _POSIX_SOURCE 1 /∗ POSIX compliant source ∗/

Functions

- void rs485_set_interface (char ∗ifname)

  RS485_Set_Interface rs485 interface name.
- const char ∗ rs485_interface (void)
- uint32_t rs485_get_baud_rate (void)
- bool rs485_set_baud_rate (uint32_t baud)

  RS485_Set_Baud_Rate set the rs485 buad rate.

- int rs485_send_handle_frame (volatile struct mstp_port_handle ∗mstp_port)

  rs485_send_handle_frame rs485 bus package a send frame, and send the package to bus.
- int rs485_recv_handle_frame (volatile struct mstp_port_handle ∗mstp_port)

  rs485_recv_handle_frame rs485 bus receive a frame, and call process these data.
- void rs485_cleanup (void)

  RS485_Cleanup The rs485 initaialize fail, have clean.
- void rs485_initialize (void)

  RS485_Initialize.

Variables

- static int RS485_Handle = -1
- static unsigned int RS485_Baud = B38400
- static char ∗ RS485_Port_Name = "/dev/ttyS0"
- static struct termios RS485_oldtio
- static struct serial_struct RS485_oldserial
- static bool RS485_SpecBaud = false

7.45.1 Macro Definition Documentation

7.45.1.1 #define _POSIX_SOURCE 1 /∗ POSIX compliant source ∗/

Definition at line 82 of file rs485.c.

7.45.1.2 #define RS485_DEBUG (1)

Definition at line 49 of file rs485.c.

7.45.1.3 #define RS485MOD 0

Definition at line 72 of file rs485.c.

7.45.2 Function Documentation

7.45.2.1 uint32_t rs485_get_baud_rate (void)

Definition at line 117 of file rs485.c.

7.45.2.2 const char∗ rs485_interface (void)

Definition at line 105 of file rs485.c.

7.45.3 Variable Documentation

7.45.3.1 unsigned int RS485_Baud = B38400 [static]

Definition at line 63 of file rs485.c.

7.45.3.2 int RS485_Handle = -1 `[static]`

Definition at line 60 of file rs485.c.

7.45.3.3 struct serial_struct RS485_oldserial `[static]`

Definition at line 77 of file rs485.c.

7.45.3.4 struct termios RS485_oldtio `[static]`

Definition at line 75 of file rs485.c.

7.45.3.5 char∗ RS485_Port_Name = "/dev/ttyS0" `[static]`

Definition at line 68 of file rs485.c.

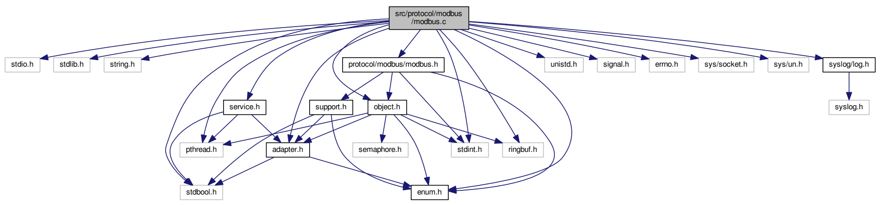
7.45.3.6 bool RS485_SpecBaud = false `[static]`

Definition at line 79 of file rs485.c.

7.46 src/protocol/modbus/modbus.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <stdint.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <pthread.h>
#include "service.h"
#include "syslog/log.h"
#include "adapter.h"
#include "enum.h"
#include "ringbuf.h"
#include "protocol/modbus/modbus.h"
#include "object.h"
```

Include dependency graph for modbus.c:



Macros

- #define RS485_MODBUS_MTU (512)

Functions

- void ∗ modbus_work_thread_function (void ∗arg)

  modbus_work_thread_function The modbus work thread
- int modbus_service_init (object_management_t ∗object)

  modbus_service_init The modbus interface intialize.
- void modbus_service_deinit (object_management_t ∗object)

  modbus_service_deinit clean the modbus service, The haved called by thread have exit.

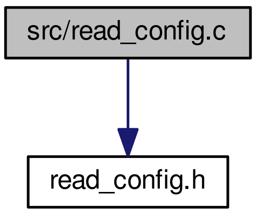
7.46.1 Macro Definition Documentation

7.46.1.1 #define RS485_MODBUS_MTU (512)

Definition at line 44 of file modbus.c.

7.47 src/read_config.c File Reference

```
#include "read_config.h"
```
Include dependency graph for read_config.c:



Variables

- int glb_config_general_work_queue_depth = 256
- int glb_config_general_work_package_mtu = 512
- int glb_config_bacnet_work_queue_depth = 128
- int glb_config_modbus_work_queue_depth = 256
- int glb_config_adapter_message_queue_depth = 32
- int glb_config_bacnet_object_instance = 10086
- int glb_config_client_max_numbers = 10

7.47.1 Variable Documentation

7.47.1.1 int glb_config_adapter_message_queue_depth = 32

Definition at line 32 of file read_config.c.

7.47.1.2 int glb_config_bacnet_object_instance = 10086

Definition at line 35 of file read_config.c.

7.47.1.3 int glb_config_bacnet_work_queue_depth = 128

Definition at line 26 of file read_config.c.

7.47.1.4 int glb_config_client_max_numbers = 10

Definition at line 37 of file read_config.c.

7.47.1.5 int glb_config_general_work_package_mtu = 512

Definition at line 23 of file read_config.c.

7.47.1.6 int glb_config_general_work_queue_depth = 256

Definition at line 21 of file read_config.c.

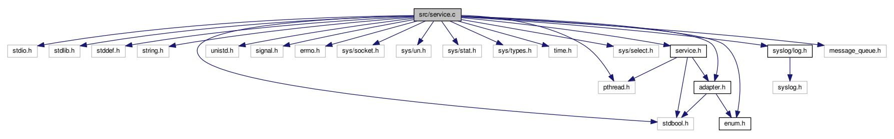
7.47.1.7 int glb_config_modbus_work_queue_depth = 256

Definition at line 29 of file read_config.c.

7.48 src/service.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stddef.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <sys/socket.h>
#include <sys/un.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>
#include <pthread.h>
#include <sys/select.h>
#include "service.h"
#include "syslog/log.h"
#include "adapter.h"
#include "enum.h"
#include "message_queue.h"
```

Include dependency graph for service.c:



Data Structures

- struct client_t

Macros

- #define RECEIVE_BUFFER_LENGTH (2048)
- #define RS485_UNIX_DOMAIN_PATH "/home/user/bin/rs485d/rs485_unix_domain_service"
- #define NALLOC (10)

Functions

- void ∗ adapter_thread_function (void ∗arg)
- void ∗ timer_task_thread_function (void ∗arg)
- static int rs485_thread_pool_create (thread_pool_t ∗pool, int numbers)

  rs485_thread_pool_create create linux thread pool
- static void rs485_thread_pool_clean (void)

  rs485_thread_pool_clean clean the linux thread haved create
- static int rs485_service_listen (int ∗socket_fd, const char ∗unix_domain_path)

  rs485_service_create create a unix domain socket communicate, used to offer rs485 service
- void rs485_service_create_clean (void)

  rs485_service_create_clean have clean the rs485 socket communicate
- int rs485_receive_from_client (int clifd, void ∗buffer, int buffer_len)
- int rs485_send_msg_to_client (int clifd, void ∗buffer, int buffer_len)

  rs485_send_msg_to_client send The message to a client
- static int process_client_request (void ∗buf, int nread, int clifd, int UNUSED(uid))
- int send_msg_to_adapter (const adapter_t ∗adapter)

  send_msg_to_adapter send a message to self,
- static void client_alloc (void)
- static int client_add (int fd, uid_t uid)
- static void client_del (int fd)
- static int serv_accept (int listenfd, uid_t ∗uidptr)
- static int rs485_service_running (const char ∗path)

  rs485_service_running The rs485 service function, It's wait the client requests. It's block
- static void rs485_service_running_clean (void)

  rs485_service_running_clean Have clean the service running
- int rs485_service_start (void)

  rs485_service_start The rs485 service start

Variables

- int glb_config_adapter_message_queue_depth
- int glb_config_client_max_numbers
- static char receive_buffer [RECEIVE_BUFFER_LENGTH]
- static int socket_fd
- static unsigned int socket_len
- static struct sockaddr_un addr
- static char unix_domain_path [108] = {0}
- static struct message_queue adapter_message_queue
- static pthread_t adapter_thread
- static pthread_t timer_task_thread
- static thread_pool_t rs485_thread_pool [ ]
- static client_t ∗ client = NULL
- static int client_size = 0

7.48.1 Detailed Description

www.enno.com

Date

: Mar 14, 2016

Author

: chuanjiang.wong

Definition in file service.c.

7.48.2 Macro Definition Documentation

7.48.2.1 #define NALLOC (10)

Definition at line 45 of file service.c.

7.48.2.2 #define RECEIVE_BUFFER_LENGTH (2048)

define the socket receive buffer length

Definition at line 38 of file service.c.

7.48.2.3 #define RS485_UNIX_DOMAIN_PATH "/home/user/bin/rs485d/rs485_unix_domain_service"

set the unix domain sinstallocket path

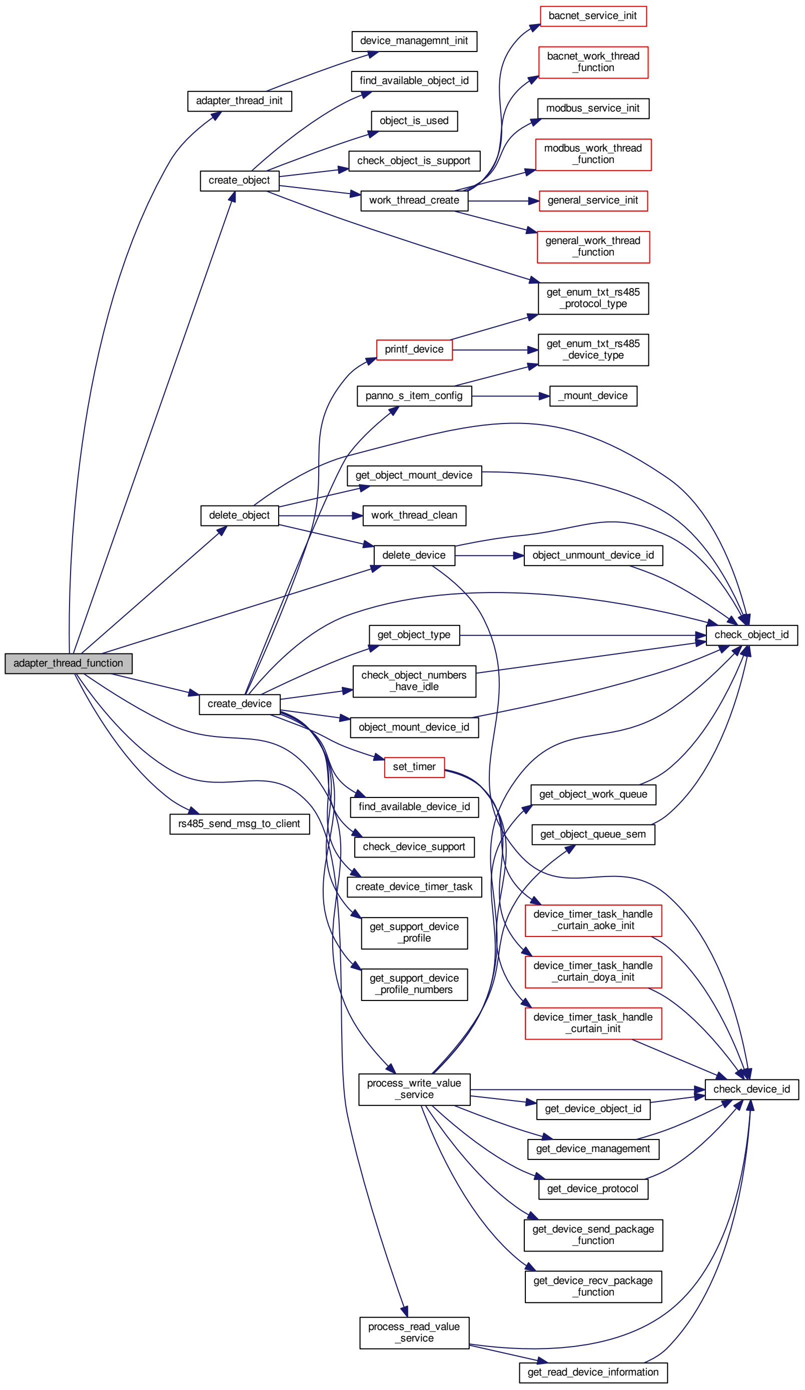
Definition at line 41 of file service.c.

7.48.3 Function Documentation

7.48.3.1 void∗ adapter_thread_function (void ∗ *arg*)

Definition at line 300 of file adapter.c.

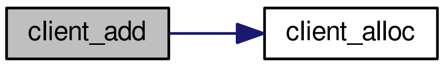
Here is the call graph for this function:



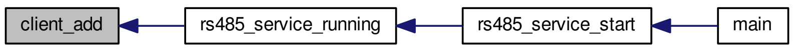
7.48.3.2 static int client_add (int *fd,* uid_t *uid*) [static]

Definition at line 315 of file service.c.

Here is the call graph for this function:



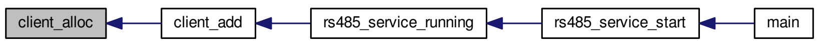
Here is the caller graph for this function:



7.48.3.3 static void client_alloc (void) [static]

Definition at line 292 of file service.c.

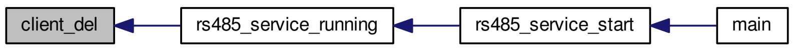
Here is the caller graph for this function:



7.48.3.4 static void client_del (int *fd*) [static]

Definition at line 340 of file service.c.

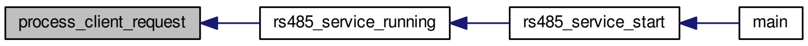
Here is the caller graph for this function:



7.48.3.5 static int process_client_request (void ∗ *buf,* int *nread,* int *clifd,* int *UNUSEDuid*) [static]

Definition at line 252 of file service.c.

Here is the caller graph for this function:



7.48.3.6 int rs485_receive_from_client (int *clifd,* void ∗ *buffer,* int *buffer_len*)

Definition at line 241 of file service.c.

7.48.3.7 static int rs485_service_listen (int ∗ *socket_fd,* const char ∗ *unix_domain_path*) [static]

rs485_service_create create a unix domain socket communicate, used to offer rs485 service

Parameters

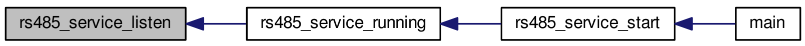
out	*socket_fd*	: The socket id, have create it.
in	*unix_domain_↩ path*	: The unix domain socket have bind a file path.

Returns

0 is success, others is fail.

Definition at line 169 of file service.c.

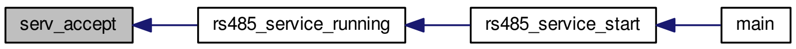
Here is the caller graph for this function:



7.48.3.8 static int serv_accept (int *listenfd,* uid_t ∗ *uidptr*) [static]

Definition at line 356 of file service.c.

Here is the caller graph for this function:



7.48.3.9 void∗ timer_task_thread_function (void ∗ *arg*)

7.48.4 Variable Documentation

7.48.4.1 struct message_queue adapter_message_queue [static]

define the adapter message queue

Definition at line 79 of file service.c.

7.48.4.2 pthread_t adapter_thread [static]

define the adapter thread

Definition at line 81 of file service.c.

7.48.4.3 struct sockaddr_un addr [static]

define the unix domain socket struct

Definition at line 75 of file service.c.

7.48.4.4 client_t∗ client = NULL [static]

Definition at line 92 of file service.c.

7.48.4.5 int client_size = 0 [static]

Definition at line 94 of file service.c.

7.48.4.6 int glb_config_adapter_message_queue_depth

Definition at line 32 of file read_config.c.

7.48.4.7 int glb_config_client_max_numbers

Definition at line 37 of file read_config.c.

7.48.4.8 char receive_buffer[RECEIVE_BUFFER_LENGTH] [static]

define the socket receive buffer

Definition at line 68 of file service.c.

7.48.4.9 thread_pool_t rs485_thread_pool[ ] [static]

Initial value:

```
=
{

    {&adapter_thread,        NULL,
      adapter_thread_function,    &adapter_message_queue,  false},
    {&timer_task_thread,     NULL,
      timer_task_thread_function, NULL,                   false},
}
```

define the rs485 service thread pool struct

Definition at line 85 of file service.c.

7.48.4.10 int socket_fd [static]

define the socket id

Definition at line 70 of file service.c.

7.48.4.11 unsigned int socket_len [static]

define the unix domain socket length

Definition at line 73 of file service.c.

7.48.4.12 pthread_t timer_task_thread [static]

define the timer task thread

Definition at line 83 of file service.c.

7.48.4.13 char unix_domain_path[108] = {0} [static]

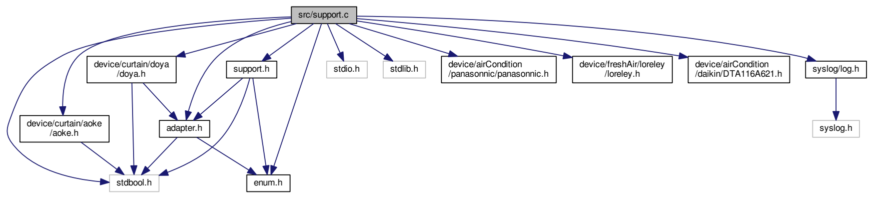
define the unix domain socket path variable , It's max length is 108

Definition at line 77 of file service.c.

7.49 src/support.c File Reference

```
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include "enum.h"
#include "adapter.h"
#include "support.h"
#include "device/airCondition/panasonnic/panasonnic.h"
#include "device/curtain/doya/doya.h"
#include "device/curtain/aoke/aoke.h"
#include "device/freshAir/loreley/loreley.h"
#include "device/airCondition/daikin/DTA116A621.h"
#include "syslog/log.h"
```

Include dependency graph for support.c:


src/support.c
device/curtain/doya/doya.h
support.h
stdio.h
stdlib.h
device/airCondition/panasonnic/panasonnic.h
device/freshAir/loreley/loreley.h
device/airCondition/daikin/DTA116A621.h
syslog/log.h
device/curtain/aoke/aoke.h
adapter.h
syslog.h
stdbool.h
enum.h

Functions

- bool check_device_support (const adapter_t ∗adatper)

  check_device_support check the device have supported by rs485 service
- struct device_profile ∗ get_support_device_profile (rs485_factory_name_enum name)

  get_support_device_profile Get the device profile, The struct device_profile
- int get_support_device_profile_numbers (rs485_factory_name_enum name)

  get_support_device_profile_numbers Get the device profile have support how many command.
- method_send get_device_send_package_function (const struct device_profile ∗profile, int profile_numbers, int command)

  get_device_send_package_function Get the device profile send package callback function
- method_recv get_device_recv_package_function (const struct device_profile ∗profile, int profile_numbers, int command)

  get_device_recv_package_function Get the device profile receive package callback function

Variables

- static struct device_profile air_condition_panasonnic []

  device_profile The panasonnic air condition device profile
- static struct device_profile curtain_doya []
- static struct device_profile curtain_aoke []
- static struct device_profile fresh_air_loreley []
- static struct device_profile air_condition_daikin_dta116a621 []

7.49.1 Variable Documentation

7.49.1.1 struct device_profile air_condition_daikin_dta116a621[ ] [static]

Definition at line 110 of file support.c.

7.49.1.2 struct device_profile air_condition_panasonnic[ ] [static]

device_profile The panasonnic air condition device profile

Definition at line 40 of file support.c.

7.49.1.3 struct device_profile curtain_aoke[ ] [static]

Initial value:

```
=
{
    {3,     RS485_CURTAIN_OPEN,
      aoke_send_package_handle,
      aoke_recv_package_handle},
    {3,     RS485_CURTAIN_CLOSE,
      aoke_send_package_handle,
      aoke_recv_package_handle},
    {3,     RS485_CURTAIN_SET_PERCENT,
      aoke_send_package_handle,
      aoke_recv_package_handle},
    {3,     RS485_CURTAIN_GET_DEVICE_INFO,
      aoke_send_package_handle,
      aoke_recv_package_handle},
}
```

Definition at line 94 of file support.c.

7.49.1.4 struct device_profile curtain_doya[ ] [static]

Initial value:

```
=
{
    {2,     RS485_DOYA_CURTAIN_OPEN,
      doya_send_package_handle,
      doya_recv_package_handle},
    {2,     RS485_DOYA_CURTAIN_CLOSE,
      doya_send_package_handle,
      doya_recv_package_handle},
    {2,     RS485_DOYA_CURTAIN_SET_PERCENT,
      doya_send_package_handle,
      doya_recv_package_handle},
    {2,     RS485_DOYA_CURTAIN_GET_DEVICE_INFO,
      doya_send_package_handle,
      doya_recv_package_handle},
}
```

Definition at line 85 of file support.c.

7.49.1.5 struct device_profile fresh_air_loreley[ ] [static]

Initial value:

```
=
{
    {1,     RS485_FRESH_AIR_AUTO_ON,
      loreley_send_package_handle,
      loreley_recv_package_handle},
    {1,     RS485_FRESH_AIR_AUTO_OFF,
      loreley_send_package_handle,
      loreley_recv_package_handle},
    {1,     RS485_FRESH_AIR_GET_DEVICE_INFO,
      loreley_send_package_handle,
      loreley_recv_package_handle},
}
```

Definition at line 102 of file support.c.

7.50 src/syslog/log.c File Reference

7.50.1 Detailed Description

www.enno.com

Date

: Mar 15, 2016

Author

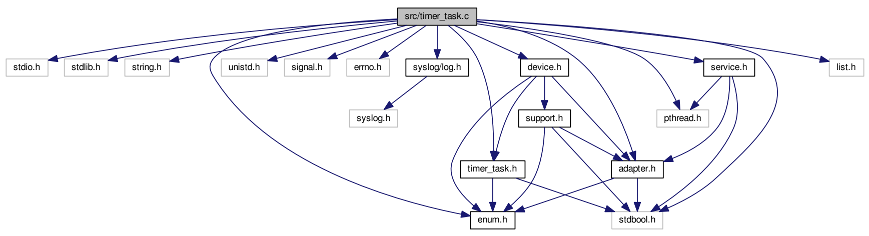
: wong

Definition in file log.c.

7.51 src/timer_task.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>
#include <unistd.h>
#include <signal.h>
#include <errno.h>
#include <pthread.h>
#include "timer_task.h"
#include "syslog/log.h"
#include "enum.h"
#include "device.h"
#include "adapter.h"
#include "service.h"
#include "list.h"
```

Include dependency graph for timer_task.c:



Macros

- #define SYSTEM_TIMER_TICK_SECOND (10)

Functions

- static int timer_task_init (void)

  timer_task_init timer task initial
- void ∗ timer_task_thread_function (void ∗UNUSED(arg))
- int create_device_timer_task (timer_task_t ∗task)

  create_deivce_timer_task create a device timer task , The timer task min tick is 10 second
- int delete_device_timer_task (timer_task_t ∗task)

  delete_device_timer_task delete a device timer task from The timer list.
- int device_timer_task_handle_demo (int device_id, int command)

  device_timer_task_handle_demo timer task handle fucntion demo
- int device_timer_task_handle_curtain_init (int device_id, int UNUSED(command))
- int device_timer_task_handle_curtain_aoke_init (int device_id, int UNUSED(command))
- int device_timer_task_handle_curtain_doya_init (int device_id, int UNUSED(command))

Variables

- static list_t ∗ timer_task_list = NULL
- static list_iterator_t ∗ timer_task_list_iterator = NULL
- static int timer_task_thread_status = TIMER_TASK_THREAD_STATUS_START
- static pthread_mutex_t timer_task_lock

7.51.1 Detailed Description

www.enno.com

Date

: Mar 14, 2016

Author

: chuanjiang.wong

Definition in file timer_task.c.

7.51.2 Macro Definition Documentation

7.51.2.1 #define SYSTEM_TIMER_TICK_SECOND (10)

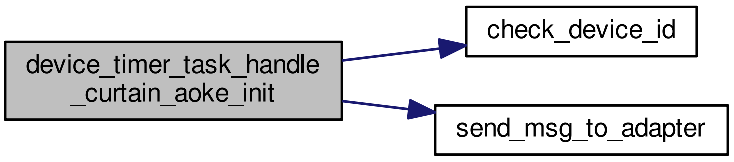
Definition at line 34 of file timer_task.c.

7.51.3 Function Documentation

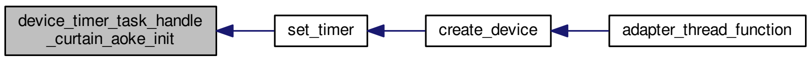
7.51.3.1 int device_timer_task_handle_curtain_aoke_init (int *device_id,* int *UNUSEDcommand*)

Definition at line 273 of file timer_task.c.

Here is the call graph for this function:



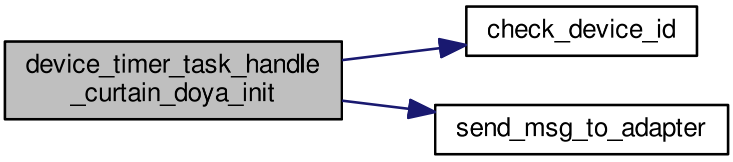
Here is the caller graph for this function:



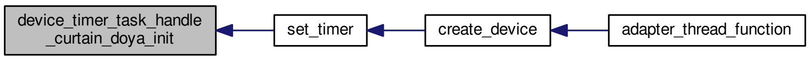
7.51.3.2 int device_timer_task_handle_curtain_doya_init (int *device_id,* int *UNUSEDcommand*)

Definition at line 303 of file timer_task.c.

Here is the call graph for this function:



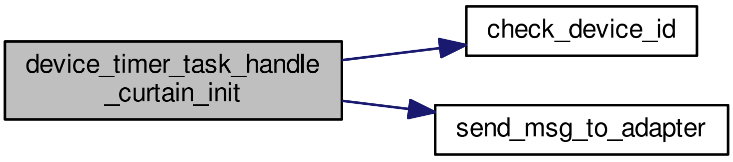
Here is the caller graph for this function:



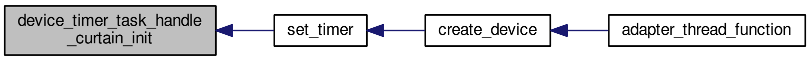
7.51.3.3 int device_timer_task_handle_curtain_init (int *device_id*, int *UNUSEDcommand*)

Definition at line 243 of file timer_task.c.

Here is the call graph for this function:



Here is the caller graph for this function:



7.51.3.4 static int timer_task_init (void) [static]

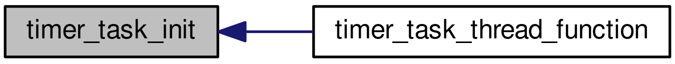
timer_task_init timer task initial

Returns

, 0 is success, others is fail.

Definition at line 58 of file timer_task.c.

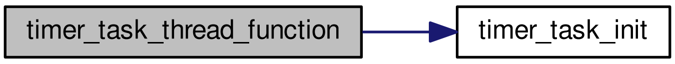
Here is the caller graph for this function:



7.51.3.5 void∗ timer_task_thread_function (void ∗ *UNUSEDarg*)

Definition at line 90 of file timer_task.c.

Here is the call graph for this function:



7.51.4 Variable Documentation

7.51.4.1 list_t∗ timer_task_list = NULL [static]

define the timer managemnt list

Definition at line 39 of file timer_task.c.

7.51.4.2 list_iterator_t∗ timer_task_list_iterator = NULL [static]

define the list iterate

Definition at line 41 of file timer_task.c.

7.51.4.3 pthread_mutex_t timer_task_lock [static]

define a mutex to used to add and delete list data

Definition at line 45 of file timer_task.c.

7.51.4.4 int timer_task_thread_status = TIMER_TASK_THREAD_STATUS_START [static]

define the timer thread status

Definition at line 43 of file timer_task.c.

Index

ADAPTER_THREAD_STATUS_INIT
    enum.h, 153
ADAPTER_THREAD_STATUS_RUNNING
    enum.h, 153
ADAPTER_THREAD_STATUS_START
    enum.h, 153
ADAPTER_THREAD_STATUS_STOP
    enum.h, 153
ADAPTER_THREAD_STATUS_UNKNOWN
    enum.h, 153
AI_COMPRESSOR_FREQUENCY
    york.c, 218
AI_INDOOR_HUMIDITY
    york.c, 218
AI_INDOOR_TEMPERATURE
    york.c, 218
AI_NET_CLEAR_TIME_LEFT
    york.c, 218
AI_OUTDOOR_TEMPERATURE
    york.c, 218
AI_SET_HUMIDITY
    york.c, 218
AI_SET_TEMPERATURE
    york.c, 218
AV_SET_HUMIDUTY
    york.c, 218
AV_SET_TEMPERATURE
    york.c, 218
Adapter management, 19
arg
    bacnet, 104

BI_CENTRAL_CONONLY_STATUS
    york.c, 218
BI_COMPRESSOR_RUNNING_STATUS
    york.c, 218
BI_COOL_ONLY_STATUS
    york.c, 218
BI_DEFROST
    york.c, 218
BI_ELECTRICAL_HEAT_STATUS
    york.c, 218
BI_ERROR_RESET_STATUS
    york.c, 218
BI_FIX_RUN_STATUS
    york.c, 218
BI_HEALTH_AIR_STATUS
    york.c, 218
BI_HOT_WATER_STATUS
    york.c, 218
BI_NET_CLEAR_RESET_STATUS
    york.c, 218
BI_NEW_AIR_STATUS
    york.c, 218
BI_ON_OFF_STATUS
    york.c, 218
BI_SAVING_STATUS
    york.c, 218
BI_SLEEP_STATUS
    york.c, 218
bacnet, 103
    arg, 104
    command, 104
    value, 104

COMMAND_CLOSE
    doya.c, 228
COMMAND_DELETE
    doya.c, 228
COMMAND_OPEN
    doya.c, 228
COMMAND_PERCENT
    doya.c, 228
COMMAND_REFACTORY
    doya.c, 228
COMMAND_STOP
    doya.c, 228
cmd
    package, 131
command
    bacnet, 104
    package, 131
Curtain, 17

data
    package, 131
Device, 15
Device management, 40
Device register management, 94
doya.c
    COMMAND_CLOSE, 228
    COMMAND_DELETE, 228
    COMMAND_OPEN, 228
    COMMAND_PERCENT, 228
    COMMAND_REFACTORY, 228
    COMMAND_STOP, 228
    RO_PERCENT, 228
    RO_VERSION, 228
    RW_DIRECTION, 228
    RW_HANDLE, 228

RW_SWITCH_ACTIVE, 228
RW_SWITCH_PASSIVE, 228
WO_ADDR, 228

enum.h
ADAPTER_THREAD_STATUS_INIT, 153
ADAPTER_THREAD_STATUS_RUNNING, 153
ADAPTER_THREAD_STATUS_START, 153
ADAPTER_THREAD_STATUS_STOP, 153
ADAPTER_THREAD_STATUS_UNKNOWN, 153
MODBUS_FUNCTION_CODE_DO_NOTHING, 153
MODBUS_FUNCTION_CODE_READ_MULTIP↩LE_COILS, 153
MODBUS_FUNCTION_CODE_READ_MULTIP↩LE_REGISTERS, 153
MODBUS_FUNCTION_CODE_READ_SIGNLE↩_COIL, 153
MODBUS_FUNCTION_CODE_READ_SIGNLE↩_REGISTER, 153
MODBUS_FUNCTION_CODE_WRITE_MULTIP↩LE_COILS, 153
MODBUS_FUNCTION_CODE_WRITE_MULTIP↩LE_REGISTERS, 153
MODBUS_FUNCTION_CODE_WRITE_SIGNLE↩_COIL, 153
MODBUS_FUNCTION_CODE_WRITE_SIGNLE↩_REGISTER, 153
OBJECT_THREAD_STATUS_INIT, 153
OBJECT_THREAD_STATUS_RUNNING, 153
OBJECT_THREAD_STATUS_START, 153
OBJECT_THREAD_STATUS_STOP, 153
OBJECT_THREAD_STATUS_UNKNOWN, 153
RS485_AIR_FAN, 154
RS485_AIR_FAN_AUTO, 154
RS485_AIR_FAN_HIGH, 154
RS485_AIR_FAN_LOW, 154
RS485_AIR_FAN_MIDDLE, 154
RS485_AIR_GET_DEVICE_INFO, 154
RS485_AIR_MODE, 154
RS485_AIR_MODE_AUTOING, 154
RS485_AIR_MODE_COOLING, 154
RS485_AIR_MODE_DRYING, 154
RS485_AIR_MODE_FANING, 154
RS485_AIR_MODE_HEATING, 154
RS485_AIR_OFF, 154
RS485_AIR_ON, 154
RS485_AIR_RESTART, 154
RS485_AIR_SET_TEMP, 153
RS485_AIR_SET_TEMP_18, 153
RS485_AIR_SET_TEMP_19, 153
RS485_AIR_SET_TEMP_20, 153
RS485_AIR_SET_TEMP_21, 153
RS485_AIR_SET_TEMP_22, 153
RS485_AIR_SET_TEMP_23, 153
RS485_AIR_SET_TEMP_24, 154
RS485_AIR_SET_TEMP_25, 154
RS485_AIR_SET_TEMP_26, 154
RS485_AIR_SET_TEMP_27, 154
RS485_AIR_SET_TEMP_28, 154
RS485_AIR_SET_TEMP_29, 154
RS485_AIR_SET_TEMP_30, 154
RS485_AIR_SWING, 154
RS485_AIR_SWING_AUTO, 154
RS485_AIR_SWING_LEFT_RIGHT, 154
RS485_AIR_SWING_UP_DOWN, 154
RS485_AIR_SWING_UP_DOWN_LEFT_RIGHT, 154
RS485_AIR_SWITCH, 154
RS485_AOKE_CURTAIN_CLOSE, 158
RS485_AOKE_CURTAIN_GET_DEVICE_INFO, 158
RS485_AOKE_CURTAIN_OPEN, 158
RS485_AOKE_CURTAIN_RESET, 158
RS485_AOKE_CURTAIN_SET_PERCENT, 158
RS485_CURTAIN, 154
RS485_CURTAIN_CLOSE, 154
RS485_CURTAIN_GET_DEVICE_INFO, 154
RS485_CURTAIN_OPEN, 154
RS485_CURTAIN_RESET, 154
RS485_CURTAIN_SET_PERCENT, 154
RS485_DEVICE_TYPE_AIR_CONDITION, 155
RS485_DEVICE_TYPE_CURTAIN, 155
RS485_DEVICE_TYPE_FRESH_AIR, 155
RS485_DEVICE_TYPE_UNKNOWN, 155
RS485_DOYA_CURTAIN_CLOSE, 158
RS485_DOYA_CURTAIN_GET_DEVICE_INFO, 158
RS485_DOYA_CURTAIN_OPEN, 158
RS485_DOYA_CURTAIN_RESET, 158
RS485_DOYA_CURTAIN_SET_PERCENT, 158
RS485_FACTORY_AOKE, 155
RS485_FACTORY_DAIKIN_DTA116A621, 155
RS485_FACTORY_DOYA, 155
RS485_FACTORY_LORELEY, 155
RS485_FACTORY_PANASONNIC, 155
RS485_FACTORY_UNKNOWN, 155
RS485_FACTORY_YORK, 155
RS485_FRESH_AIR, 154
RS485_FRESH_AIR_AUTO_OFF, 154
RS485_FRESH_AIR_AUTO_ON, 154
RS485_FRESH_AIR_GET_DEVICE_INFO, 154
RS485_FRESH_AIR_RESET, 154
RS485_LORELEY_FRESH_AIR_AUTO_OFF, 158
RS485_LORELEY_FRESH_AIR_AUTO_ON, 158
RS485_LORELEY_FRESH_AIR_GET_DEVICE↩_INFO, 158
RS485_LORELEY_FRESH_AIR_RESET, 158
RS485_PANASONNIC_AIR_FAN_AUTO, 156
RS485_PANASONNIC_AIR_FAN_HIGH, 156
RS485_PANASONNIC_AIR_FAN_LOW, 156
RS485_PANASONNIC_AIR_FAN_MIDDLE, 156
RS485_PANASONNIC_AIR_FAN_MOST, 156
RS485_PANASONNIC_AIR_FAN_MUTE, 156
RS485_PANASONNIC_AIR_GET_DEVICE_INF↩O, 156

RS485_PANASONNIC_AIR_MODE_AUTOING, 156
RS485_PANASONNIC_AIR_MODE_COOLING, 156
RS485_PANASONNIC_AIR_MODE_DRYING, 156
RS485_PANASONNIC_AIR_MODE_FANING, 156
RS485_PANASONNIC_AIR_MODE_HEATING, 156
RS485_PANASONNIC_AIR_OFF, 156
RS485_PANASONNIC_AIR_ON, 156
RS485_PANASONNIC_AIR_RESET, 156
RS485_PANASONNIC_AIR_SET_TEMP_16, 155
RS485_PANASONNIC_AIR_SET_TEMP_17, 155
RS485_PANASONNIC_AIR_SET_TEMP_18, 155
RS485_PANASONNIC_AIR_SET_TEMP_19, 155
RS485_PANASONNIC_AIR_SET_TEMP_20, 155
RS485_PANASONNIC_AIR_SET_TEMP_21, 155
RS485_PANASONNIC_AIR_SET_TEMP_22, 155
RS485_PANASONNIC_AIR_SET_TEMP_23, 155
RS485_PANASONNIC_AIR_SET_TEMP_24, 155
RS485_PANASONNIC_AIR_SET_TEMP_25, 155
RS485_PANASONNIC_AIR_SET_TEMP_26, 155
RS485_PANASONNIC_AIR_SET_TEMP_27, 155
RS485_PANASONNIC_AIR_SET_TEMP_28, 155
RS485_PANASONNIC_AIR_SET_TEMP_29, 155
RS485_PANASONNIC_AIR_SET_TEMP_30, 155
RS485_PANASONNIC_AIR_SWING_AUTO, 155
RS485_PANASONNIC_AIR_SWING_HAND1, 156
RS485_PANASONNIC_AIR_SWING_HAND2, 156
RS485_PANASONNIC_AIR_SWING_HAND3, 156
RS485_PANASONNIC_AIR_SWING_HAND4, 156
RS485_PANASONNIC_AIR_SWING_HAND5, 156
RS485_PROTOCOL_TYPE_BACNET, 158
RS485_PROTOCOL_TYPE_GENERAL, 158
RS485_PROTOCOL_TYPE_MODBUS, 158
RS485_PROTOCOL_TYPE_UNKNOWN, 158
RS485_YORK_AIR_CENTRAL_CONTROL_ON←LY_NO, 157
RS485_YORK_AIR_CENTRAL_CONTROL_ON←LY_YES, 157
RS485_YORK_AIR_COOL_ONLY_NO, 157
RS485_YORK_AIR_COOL_ONLY_YES, 157
RS485_YORK_AIR_DEFROST_NO, 157
RS485_YORK_AIR_DEFROST_YES, 157
RS485_YORK_AIR_ELECTRICAL_HEAT_NO, 157
RS485_YORK_AIR_ELECTRICAL_HEAT_YES, 157
RS485_YORK_AIR_ERR_RESET_NO, 157
RS485_YORK_AIR_ERR_RESET_YES, 157
RS485_YORK_AIR_FAN_AUTO, 157
RS485_YORK_AIR_FAN_HIGH, 157
RS485_YORK_AIR_FAN_LOW, 157
RS485_YORK_AIR_FAN_MIDDLE, 157
RS485_YORK_AIR_FIX_RUN_NO, 157
RS485_YORK_AIR_FIX_RUN_YES, 157
RS485_YORK_AIR_GET_DEVICE_INFO, 157
RS485_YORK_AIR_HEALTH_AIR_NO, 157
RS485_YORK_AIR_HEALTH_AIR_YES, 157
RS485_YORK_AIR_HOME_LEFT_NO, 157
RS485_YORK_AIR_HOME_LEFT_YES, 157
RS485_YORK_AIR_HOT_WATER_NO, 157
RS485_YORK_AIR_HOT_WATER_YES, 157
RS485_YORK_AIR_MODE_AUTOING, 157
RS485_YORK_AIR_MODE_COOLING, 157
RS485_YORK_AIR_MODE_DRYING, 157
RS485_YORK_AIR_MODE_FANING, 157
RS485_YORK_AIR_MODE_HEATING, 157
RS485_YORK_AIR_NET_RESET_NO, 157
RS485_YORK_AIR_NET_RESET_YES, 157
RS485_YORK_AIR_OFF, 157
RS485_YORK_AIR_ON, 157
RS485_YORK_AIR_SAVING_NO, 157
RS485_YORK_AIR_SAVING_YES, 157
RS485_YORK_AIR_SET_HUMIDITY, 156
RS485_YORK_AIR_SET_TEMP_18, 156
RS485_YORK_AIR_SET_TEMP_19, 156
RS485_YORK_AIR_SET_TEMP_20, 156
RS485_YORK_AIR_SET_TEMP_21, 156
RS485_YORK_AIR_SET_TEMP_22, 156
RS485_YORK_AIR_SET_TEMP_23, 156
RS485_YORK_AIR_SET_TEMP_24, 156
RS485_YORK_AIR_SET_TEMP_25, 156
RS485_YORK_AIR_SET_TEMP_26, 156
RS485_YORK_AIR_SET_TEMP_27, 156
RS485_YORK_AIR_SET_TEMP_28, 156
RS485_YORK_AIR_SET_TEMP_29, 156
RS485_YORK_AIR_SET_TEMP_30, 156
RS485_YORK_AIR_SET_TEMP_31, 156
RS485_YORK_AIR_SET_TEMP_32, 156
RS485_YORK_AIR_SLEEP_NO, 157
RS485_YORK_AIR_SLEEP_YES, 157
RS485_YORK_AIR_SWING_AUTO, 156
RS485_YORK_AIR_SWING_LEFT_RIGHT, 157
RS485_YORK_AIR_SWING_UP_DOWN, 156
RS485_YORK_AIR_SWING_UP_DOWN_LEFT←_RIGHT, 157
SERVICE_CREATE_RS485_OBJECT, 159
SERVICE_DELETE_RS485_OBJECT, 159
SERVICE_MOUNT_DEVICE_TO_OBJECT, 159
SERVICE_READ_VALUE_FROM_DEVICE, 159
SERVICE_SYSTEM_UPDATE_START, 159
SERVICE_SYSTEM_UPDATE_STOP, 159
SERVICE_UNKNOWN, 159
SERVICE_UNMOUNT_DEVICE_FROM_OBJE←CT, 159
SERVICE_WRITE_VALUE_TO_DEVICE, 159
TIMER_TASK_THREAD_STATUS_ADDING, 159
TIMER_TASK_THREAD_STATUS_DELETEING, 159
TIMER_TASK_THREAD_STATUS_INIT, 159
TIMER_TASK_THREAD_STATUS_RUNNING, 159
TIMER_TASK_THREAD_STATUS_START, 159
TIMER_TASK_THREAD_STATUS_STOP, 159

TIMER_TASK_THREAD_STATUS_UNKNOWN, 159

General, 14
General interface, 79

Item management, 58

MI_COMMUNICATION_STATUS
york.c, 218
MI_ERROR
york.c, 218
MI_FAN
york.c, 218
MI_INDOOR_STYLE
york.c, 218
MI_LOCAL_SET
york.c, 218
MI_MODE
york.c, 218
MI_RESERVED
york.c, 218
MI_SWING
york.c, 218
MI_VENTILATION
york.c, 218
MODBUS_FUNCTION_CODE_DO_NOTHING
enum.h, 153
MODBUS_FUNCTION_CODE_READ_MULTIPLE_C←OILS
enum.h, 153
MODBUS_FUNCTION_CODE_READ_MULTIPLE_R←EGISTERS
enum.h, 153
MODBUS_FUNCTION_CODE_READ_SIGNLE_COIL
enum.h, 153
MODBUS_FUNCTION_CODE_READ_SIGNLE_REG←ISTER
enum.h, 153
MODBUS_FUNCTION_CODE_WRITE_MULTIPLE_←COILS
enum.h, 153
MODBUS_FUNCTION_CODE_WRITE_MULTIPLE_←REGISTERS
enum.h, 153
MODBUS_FUNCTION_CODE_WRITE_SIGNLE_CO←IL
enum.h, 153
MODBUS_FUNCTION_CODE_WRITE_SIGNLE_RE←GISTER
enum.h, 153
Management, 9
Modbus, 13

OBJECT_THREAD_STATUS_INIT
enum.h, 153
OBJECT_THREAD_STATUS_RUNNING
enum.h, 153
OBJECT_THREAD_STATUS_START
enum.h, 153
OBJECT_THREAD_STATUS_STOP
enum.h, 153
OBJECT_THREAD_STATUS_UNKNOWN
enum.h, 153
Object management, 60

package, 130
cmd, 131
command, 131
data, 131
Protocol, 11

RO_PERCENT
doya.c, 228
RO_VERSION
doya.c, 228
RS485_AIR_FAN
enum.h, 154
RS485_AIR_FAN_AUTO
enum.h, 154
RS485_AIR_FAN_HIGH
enum.h, 154
RS485_AIR_FAN_LOW
enum.h, 154
RS485_AIR_FAN_MIDDLE
enum.h, 154
RS485_AIR_GET_DEVICE_INFO
enum.h, 154
RS485_AIR_MODE
enum.h, 154
RS485_AIR_MODE_AUTOING
enum.h, 154
RS485_AIR_MODE_COOLING
enum.h, 154
RS485_AIR_MODE_DRYING
enum.h, 154
RS485_AIR_MODE_FANING
enum.h, 154
RS485_AIR_MODE_HEATING
enum.h, 154
RS485_AIR_OFF
enum.h, 154
RS485_AIR_ON
enum.h, 154
RS485_AIR_RESTART
enum.h, 154
RS485_AIR_SET_TEMP
enum.h, 153
RS485_AIR_SET_TEMP_18
enum.h, 153
RS485_AIR_SET_TEMP_19
enum.h, 153
RS485_AIR_SET_TEMP_20
enum.h, 153
RS485_AIR_SET_TEMP_21
enum.h, 153
RS485_AIR_SET_TEMP_22
enum.h, 153

RS485_AIR_SET_TEMP_23
enum.h, 153
RS485_AIR_SET_TEMP_24
enum.h, 154
RS485_AIR_SET_TEMP_25
enum.h, 154
RS485_AIR_SET_TEMP_26
enum.h, 154
RS485_AIR_SET_TEMP_27
enum.h, 154
RS485_AIR_SET_TEMP_28
enum.h, 154
RS485_AIR_SET_TEMP_29
enum.h, 154
RS485_AIR_SET_TEMP_30
enum.h, 154
RS485_AIR_SWING
enum.h, 154
RS485_AIR_SWING_AUTO
enum.h, 154
RS485_AIR_SWING_LEFT_RIGHT
enum.h, 154
RS485_AIR_SWING_UP_DOWN
enum.h, 154
RS485_AIR_SWING_UP_DOWN_LEFT_RIGHT
enum.h, 154
RS485_AIR_SWITCH
enum.h, 154
RS485_AOKE_CURTAIN_CLOSE
enum.h, 158
RS485_AOKE_CURTAIN_GET_DEVICE_INFO
enum.h, 158
RS485_AOKE_CURTAIN_OPEN
enum.h, 158
RS485_AOKE_CURTAIN_RESET
enum.h, 158
RS485_AOKE_CURTAIN_SET_PERCENT
enum.h, 158
RS485_CURTAIN
enum.h, 154
RS485_CURTAIN_CLOSE
enum.h, 154
RS485_CURTAIN_GET_DEVICE_INFO
enum.h, 154
RS485_CURTAIN_OPEN
enum.h, 154
RS485_CURTAIN_RESET
enum.h, 154
RS485_CURTAIN_SET_PERCENT
enum.h, 154
RS485_DEVICE_TYPE_AIR_CONDITION
enum.h, 155
RS485_DEVICE_TYPE_CURTAIN
enum.h, 155
RS485_DEVICE_TYPE_FRESH_AIR
enum.h, 155
RS485_DEVICE_TYPE_UNKNOWN
enum.h, 155
RS485_DOYA_CURTAIN_CLOSE
enum.h, 158
RS485_DOYA_CURTAIN_GET_DEVICE_INFO
enum.h, 158
RS485_DOYA_CURTAIN_OPEN
enum.h, 158
RS485_DOYA_CURTAIN_RESET
enum.h, 158
RS485_DOYA_CURTAIN_SET_PERCENT
enum.h, 158
RS485_FACTORY_AOKE
enum.h, 155
RS485_FACTORY_DAIKIN_DTA116A621
enum.h, 155
RS485_FACTORY_DOYA
enum.h, 155
RS485_FACTORY_LORELEY
enum.h, 155
RS485_FACTORY_PANASONNIC
enum.h, 155
RS485_FACTORY_UNKNOWN
enum.h, 155
RS485_FACTORY_YORK
enum.h, 155
RS485_FRESH_AIR
enum.h, 154
RS485_FRESH_AIR_AUTO_OFF
enum.h, 154
RS485_FRESH_AIR_AUTO_ON
enum.h, 154
RS485_FRESH_AIR_GET_DEVICE_INFO
enum.h, 154
RS485_FRESH_AIR_RESET
enum.h, 154
RS485_LORELEY_FRESH_AIR_AUTO_OFF
enum.h, 158
RS485_LORELEY_FRESH_AIR_AUTO_ON
enum.h, 158
RS485_LORELEY_FRESH_AIR_GET_DEVICE_INFO
enum.h, 158
RS485_LORELEY_FRESH_AIR_RESET
enum.h, 158
RS485_PANASONNIC_AIR_FAN_AUTO
enum.h, 156
RS485_PANASONNIC_AIR_FAN_HIGH
enum.h, 156
RS485_PANASONNIC_AIR_FAN_LOW
enum.h, 156
RS485_PANASONNIC_AIR_FAN_MIDDLE
enum.h, 156
RS485_PANASONNIC_AIR_FAN_MOST
enum.h, 156
RS485_PANASONNIC_AIR_FAN_MUTE
enum.h, 156
RS485_PANASONNIC_AIR_GET_DEVICE_INFO
enum.h, 156
RS485_PANASONNIC_AIR_MODE_AUTOING
enum.h, 156

RS485_PANASONNIC_AIR_MODE_COOLING
enum.h, 156
RS485_PANASONNIC_AIR_MODE_DRYING
enum.h, 156
RS485_PANASONNIC_AIR_MODE_FANING
enum.h, 156
RS485_PANASONNIC_AIR_MODE_HEATING
enum.h, 156
RS485_PANASONNIC_AIR_OFF
enum.h, 156
RS485_PANASONNIC_AIR_ON
enum.h, 156
RS485_PANASONNIC_AIR_RESET
enum.h, 156
RS485_PANASONNIC_AIR_SET_TEMP_16
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_17
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_18
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_19
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_20
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_21
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_22
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_23
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_24
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_25
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_26
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_27
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_28
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_29
enum.h, 155
RS485_PANASONNIC_AIR_SET_TEMP_30
enum.h, 155
RS485_PANASONNIC_AIR_SWING_AUTO
enum.h, 155
RS485_PANASONNIC_AIR_SWING_HAND1
enum.h, 156
RS485_PANASONNIC_AIR_SWING_HAND2
enum.h, 156
RS485_PANASONNIC_AIR_SWING_HAND3
enum.h, 156
RS485_PANASONNIC_AIR_SWING_HAND4
enum.h, 156
RS485_PANASONNIC_AIR_SWING_HAND5
enum.h, 156
RS485_PROTOCOL_TYPE_BACNET
enum.h, 158
RS485_PROTOCOL_TYPE_GENERAL
enum.h, 158
RS485_PROTOCOL_TYPE_MODBUS
enum.h, 158
RS485_PROTOCOL_TYPE_UNKNOWN
enum.h, 158
RS485_YORK_AIR_CENTRAL_CONTROL_ONLY_↩
NO
enum.h, 157
RS485_YORK_AIR_CENTRAL_CONTROL_ONLY_Y↩
ES
enum.h, 157
RS485_YORK_AIR_COOL_ONLY_NO
enum.h, 157
RS485_YORK_AIR_COOL_ONLY_YES
enum.h, 157
RS485_YORK_AIR_DEFROST_NO
enum.h, 157
RS485_YORK_AIR_DEFROST_YES
enum.h, 157
RS485_YORK_AIR_ELECTRICAL_HEAT_NO
enum.h, 157
RS485_YORK_AIR_ELECTRICAL_HEAT_YES
enum.h, 157
RS485_YORK_AIR_ERR_RESET_NO
enum.h, 157
RS485_YORK_AIR_ERR_RESET_YES
enum.h, 157
RS485_YORK_AIR_FAN_AUTO
enum.h, 157
RS485_YORK_AIR_FAN_HIGH
enum.h, 157
RS485_YORK_AIR_FAN_LOW
enum.h, 157
RS485_YORK_AIR_FAN_MIDDLE
enum.h, 157
RS485_YORK_AIR_FIX_RUN_NO
enum.h, 157
RS485_YORK_AIR_FIX_RUN_YES
enum.h, 157
RS485_YORK_AIR_GET_DEVICE_INFO
enum.h, 157
RS485_YORK_AIR_HEALTH_AIR_NO
enum.h, 157
RS485_YORK_AIR_HEALTH_AIR_YES
enum.h, 157
RS485_YORK_AIR_HOME_LEFT_NO
enum.h, 157
RS485_YORK_AIR_HOME_LEFT_YES
enum.h, 157
RS485_YORK_AIR_HOT_WATER_NO
enum.h, 157
RS485_YORK_AIR_HOT_WATER_YES
enum.h, 157
RS485_YORK_AIR_MODE_AUTOING
enum.h, 157
RS485_YORK_AIR_MODE_COOLING
enum.h, 157

RS485_YORK_AIR_MODE_DRYING
enum.h, 157
RS485_YORK_AIR_MODE_FANING
enum.h, 157
RS485_YORK_AIR_MODE_HEATING
enum.h, 157
RS485_YORK_AIR_NET_RESET_NO
enum.h, 157
RS485_YORK_AIR_NET_RESET_YES
enum.h, 157
RS485_YORK_AIR_OFF
enum.h, 157
RS485_YORK_AIR_ON
enum.h, 157
RS485_YORK_AIR_SAVING_NO
enum.h, 157
RS485_YORK_AIR_SAVING_YES
enum.h, 157
RS485_YORK_AIR_SET_HUMIDITY
enum.h, 156
RS485_YORK_AIR_SET_TEMP_18
enum.h, 156
RS485_YORK_AIR_SET_TEMP_19
enum.h, 156
RS485_YORK_AIR_SET_TEMP_20
enum.h, 156
RS485_YORK_AIR_SET_TEMP_21
enum.h, 156
RS485_YORK_AIR_SET_TEMP_22
enum.h, 156
RS485_YORK_AIR_SET_TEMP_23
enum.h, 156
RS485_YORK_AIR_SET_TEMP_24
enum.h, 156
RS485_YORK_AIR_SET_TEMP_25
enum.h, 156
RS485_YORK_AIR_SET_TEMP_26
enum.h, 156
RS485_YORK_AIR_SET_TEMP_27
enum.h, 156
RS485_YORK_AIR_SET_TEMP_28
enum.h, 156
RS485_YORK_AIR_SET_TEMP_29
enum.h, 156
RS485_YORK_AIR_SET_TEMP_30
enum.h, 156
RS485_YORK_AIR_SET_TEMP_31
enum.h, 156
RS485_YORK_AIR_SET_TEMP_32
enum.h, 156
RS485_YORK_AIR_SLEEP_NO
enum.h, 157
RS485_YORK_AIR_SLEEP_YES
enum.h, 157
RS485_YORK_AIR_SWING_AUTO
enum.h, 156
RS485_YORK_AIR_SWING_LEFT_RIGHT
enum.h, 157
RS485_YORK_AIR_SWING_UP_DOWN
enum.h, 156
RS485_YORK_AIR_SWING_UP_DOWN_LEFT_RIG←
HT
enum.h, 157
RW_DIRECTION
doya.c, 228
RW_HANDLE
doya.c, 228
RW_SWITCH_ACTIVE
doya.c, 228
RW_SWITCH_PASSIVE
doya.c, 228

SERVICE_CREATE_RS485_OBJECT
enum.h, 159
SERVICE_DELETE_RS485_OBJECT
enum.h, 159
SERVICE_MOUNT_DEVICE_TO_OBJECT
enum.h, 159
SERVICE_READ_VALUE_FROM_DEVICE
enum.h, 159
SERVICE_SYSTEM_UPDATE_START
enum.h, 159
SERVICE_SYSTEM_UPDATE_STOP
enum.h, 159
SERVICE_UNKNOWN
enum.h, 159
SERVICE_UNMOUNT_DEVICE_FROM_OBJECT
enum.h, 159
SERVICE_WRITE_VALUE_TO_DEVICE
enum.h, 159
Service management, 89
String management, 55

TIMER_TASK_THREAD_STATUS_ADDING
enum.h, 159
TIMER_TASK_THREAD_STATUS_DELETEING
enum.h, 159
TIMER_TASK_THREAD_STATUS_INIT
enum.h, 159
TIMER_TASK_THREAD_STATUS_RUNNING
enum.h, 159
TIMER_TASK_THREAD_STATUS_START
enum.h, 159
TIMER_TASK_THREAD_STATUS_STOP
enum.h, 159
TIMER_TASK_THREAD_STATUS_UNKNOWN
enum.h, 159
Timer management, 98

value
bacnet, 104

WO_ADDR
doya.c, 228

york.c
AI_COMPRESSOR_FREQUENCY, 218

AI_INDOOR_HUMIDITY, 218
AI_INDOOR_TEMPERATURE, 218
AI_NET_CLEAR_TIME_LEFT, 218
AI_OUTDOOR_TEMPERATURE, 218
AI_SET_HUMIDITY, 218
AI_SET_TEMPERATURE, 218
AV_SET_HUMIDUTY, 218
AV_SET_TEMPERATURE, 218
BI_CENTRAL_CONONLY_STATUS, 218
BI_COMPRESSOR_RUNNING_STATUS, 218
BI_COOL_ONLY_STATUS, 218
BI_DEFROST, 218
BI_ELECTRICAL_HEAT_STATUS, 218
BI_ERROR_RESET_STATUS, 218
BI_FIX_RUN_STATUS, 218
BI_HEALTH_AIR_STATUS, 218
BI_HOT_WATER_STATUS, 218
BI_NET_CLEAR_RESET_STATUS, 218
BI_NEW_AIR_STATUS, 218
BI_ON_OFF_STATUS, 218
BI_SAVING_STATUS, 218
BI_SLEEP_STATUS, 218
MI_COMMUNICATION_STATUS, 218
MI_ERROR, 218
MI_FAN, 218
MI_INDOOR_STYLE, 218
MI_LOCAL_SET, 218
MI_MODE, 218
MI_RESERVED, 218
MI_SWING, 218
MI_VENTILATION, 218