

ezManager Library User's Manual

Revision 1.1

Sollae Systems



Sollae Systems. Co., Ltd.

- 0 -

<https://www.eztcp.com>

Contents

1	OVERVIEW	- 8 -
1.1	Overview	- 8 -
1.2	Conventions Used in This Document	- 8 -
1.3	WARNING	- 8 -
2	LIBRARY INITIALIZATION AND FINALIZATION	- 9 -
2.1	Start_Library	- 9 -
2.2	End_Library	- 9 -
3	SEARCH EZTCP	- 11 -
3.1	EzTCP_Search	- 11 -
3.2	EzTCP_Read	- 12 -
3.3	get_eztcp_count	- 14 -
3.4	get_eztcp_info_by_index	- 16 -
4	READ PARAMETERS	- 19 -
4.1	Network	- 19 -
4.1.1	get_arp	- 19 -
4.1.2	get_dhcp	- 19 -
4.1.3	get_dhcp_dns	- 20 -
4.1.4	get_pppoe	- 20 -
4.1.5	get_ip4_local_address	- 21 -
4.1.6	get_ip4_subnet_mask	- 22 -
4.1.7	get_ip4_gateway_address	- 23 -
4.1.8	get_ip4_dns_address	- 24 -
4.1.9	get_pppoe_id	- 25 -
4.1.10	get_pppoe_pwd	- 26 -
4.1.11	get_ip6	- 27 -
4.1.12	get_ip6_gua_type	- 27 -
4.1.13	get_ip6_eui_type	- 28 -
4.1.14	get_ip6_local_address	- 29 -
4.1.15	get_ip6_subnet_prefix_length	- 30 -
4.1.16	get_ip6_gateway_address	- 31 -
4.1.17	get_ip6_dns_address	- 32 -
4.2	Serial Port	- 33 -
4.2.1	get_uart_count	- 33 -
4.2.2	get_uart_min_baudrate	- 34 -
4.2.3	get_uart_max_baudrate	- 35 -
4.2.4	get_uart_serial_type	- 35 -
4.2.5	get_uart_ttl	- 36 -
4.2.6	get_uart_baudrate	- 37 -
4.2.7	get_uart_parity	- 37 -
4.2.8	get_uart_databit	- 38 -
4.2.9	get_uart_stopbit	- 39 -
4.2.10	get_uart_flow_control	- 40 -
4.2.11	get_uart_dtrdsr	- 41 -
4.2.12	get_uart_tx_delay	- 42 -
4.3	TCP/IP	- 42 -
4.3.1	get_uart_communication_mode	- 42 -
4.3.2	get_uart_peer_address	- 43 -
4.3.3	get_uart_peer_port	- 45 -
4.3.4	get_uart_local_port	- 46 -
4.3.5	get_uart_cod_tcp_server	- 47 -
4.3.6	get_uart_watermark	- 47 -

4.3.7	<i>get_uart_timeout</i>	- 49 -
4.3.8	<i>get_uart_data_frame_interval</i>	- 50 -
4.3.9	<i>get_uart_separator_length</i>	- 51 -
4.3.10	<i>get_uart_separator_type</i>	- 52 -
4.3.11	<i>get_uart_separator</i>	- 53 -
4.3.12	<i>get_uart_tcp_nodelay</i>	- 54 -
4.3.13	<i>get_uart_rfc2217</i>	- 55 -
4.3.14	<i>get_uart_protocol</i>	- 56 -
4.4	CSC-HR2.....	- 57 -
4.4.1	<i>get_csc_hr2_communication_mode</i>	- 57 -
4.4.2	<i>get_csc_hr2_id</i>	- 57 -
4.4.3	<i>get_csc_hr2_network_timeout</i>	- 58 -
4.4.4	<i>get_csc_hr2_network_threshold</i>	- 59 -
4.4.5	<i>get_csc_hr2_server_timeout</i>	- 60 -
4.4.6	<i>get_csc_hr2_server_threshold</i>	- 61 -
4.4.7	<i>get_csc_hr2_check_port</i>	- 62 -
4.4.8	<i>get_csc_hr2_first_server_address</i>	- 63 -
4.4.9	<i>get_csc_hr2_first_server_port</i>	- 64 -
4.4.10	<i>get_csc_hr2_second_server_address</i>	- 65 -
4.4.11	<i>get_csc_hr2_second_server_port</i>	- 66 -
4.5	I/O Controller.....	- 68 -
4.5.1	<i>get_io_http</i>	- 68 -
4.5.2	<i>get_io_http_port</i>	- 68 -
4.5.3	<i>get_io_html_size</i>	- 69 -
4.5.4	<i>get_io_modbus</i>	- 70 -
4.5.5	<i>get_io_input_notification</i>	- 71 -
4.5.6	<i>get_io_output_automatic_initialize</i>	- 72 -
4.5.7	<i>get_io_modbus_type</i>	- 72 -
4.5.8	<i>get_io_unit_id</i>	- 74 -
4.5.9	<i>get_io_input_address</i>	- 75 -
4.5.10	<i>get_io_output_address</i>	- 76 -
4.5.11	<i>get_io_poll_interval</i>	- 77 -
4.5.12	<i>get_io_slave_control_type</i>	- 78 -
4.5.13	<i>get_io_master_control_type</i>	- 79 -
4.5.14	<i>get_io_tcp_type</i>	- 80 -
4.5.15	<i>get_io_multi_connection_number</i>	- 81 -
4.5.16	<i>get_io_modbus_peer_address</i>	- 82 -
4.5.17	<i>get_io_modbus_peer_port</i>	- 83 -
4.5.18	<i>get_io_modbus_local_port</i>	- 84 -
4.5.19	<i>get_io_output_number</i>	- 85 -
4.5.20	<i>get_io_input_number</i>	- 86 -
4.5.21	<i>get_io_event_notification</i>	- 87 -
4.5.22	<i>get_event_notification_email</i>	- 87 -
4.5.23	<i>get_io_event_notification_port</i>	- 88 -
4.5.24	<i>get_io_input_valid_time</i>	- 90 -
4.5.25	<i>get_io_macro_type</i>	- 91 -
4.5.26	<i>get_io_macro</i>	- 92 -
4.5.27	<i>get_io_port_macro</i>	- 93 -
4.5.28	<i>get_io_macro_text</i>	- 94 -
4.5.29	<i>get_io_output_delay</i>	- 96 -
4.5.30	<i>get_io_output_initial_state</i>	- 97 -
4.5.31	<i>get_io_comment</i>	- 98 -
4.6	Wireless Network	- 100 -
4.6.1	<i>get_wlan_type</i>	- 100 -
4.6.2	<i>get_wlan_channel</i>	- 101 -
4.6.3	<i>get_wlan_ssid</i>	- 102 -
4.6.4	<i>get_wlan_antenna</i>	- 103 -

4.6.5	<i>get_wlan_phy_mode</i>	- 104 -
4.6.6	<i>get_wlan_short_preamble</i>	- 106 -
4.6.7	<i>get_wlan_short_slot</i>	- 107 -
4.6.8	<i>get_wlan_cts_protection</i>	- 108 -
4.6.9	<i>get_wlan_background_scan</i>	- 109 -
4.6.10	<i>get_wlan_encryption_type</i>	- 110 -
4.6.11	<i>get_wlan_authentication_type</i>	- 112 -
4.6.12	<i>get_wlan_wep_key_length</i>	- 114 -
4.6.13	<i>get_wlan_wep_key_index</i>	- 115 -
4.6.14	<i>get_wlan_wep_key_data_type</i>	- 116 -
4.6.15	<i>get_wlan_wep_key</i>	- 117 -
4.6.16	<i>get_wlan_max_wpa_passphrase_length</i>	- 119 -
4.6.17	<i>get_wlan_wpa_passphrase</i>	- 119 -
4.6.18	<i>get_wlan_shared_key</i>	- 121 -
4.6.19	<i>get_wlan_wpa_enterprise</i>	- 122 -
4.6.20	<i>get_wlan_wpa_enterprise_id</i>	- 123 -
4.6.21	<i>get_wlan_wpa_enterprise_pwd</i>	- 125 -
4.7	Options	- 126 -
4.7.1	<i>get_telnet</i>	- 126 -
4.7.2	<i>get_send_mac_address</i>	- 127 -
4.7.3	<i>get_ssl</i>	- 127 -
4.7.4	<i>get_ssh</i>	- 128 -
4.7.5	<i>get_ip4_address_search</i>	- 128 -
4.7.6	<i>get_remote_debug</i>	- 129 -
4.7.7	<i>get_tcp_multi_connection</i>	- 130 -
4.7.8	<i>get_power_management</i>	- 130 -
4.7.9	<i>get_comment</i>	- 131 -
4.7.10	<i>get_allowed_ezmanager</i>	- 132 -
4.7.11	<i>get_allowed_mac_address</i>	- 133 -
4.7.12	<i>get_allowed_ip4_address</i>	- 134 -
4.7.13	<i>get_allowed_ip4_network_mask_type</i>	- 135 -
4.7.14	<i>get_allowed_ip6_address</i>	- 137 -
4.7.15	<i>get_allowed_ip6_subnet_prefix_length</i>	- 138 -
4.7.16	<i>get_ip4_change_notification_type</i>	- 140 -
4.7.17	<i>get_ip4_change_notification_data_type</i>	- 141 -
4.7.18	<i>get_ip4_change_notification_interval</i>	- 142 -
4.7.19	<i>get_ip4_change_notification_peer_port</i>	- 143 -
4.7.20	<i>get_ip4_change_notification_peer_address</i>	- 144 -
4.7.21	<i>get_ip4_change_notification_ddns_id</i>	- 146 -
4.7.22	<i>get_ip4_change_notification_ddns_pwd</i>	- 147 -
5	CHANGE PARAMETERS	- 149 -
5.1	Network	- 149 -
5.1.1	<i>set_arp</i>	- 149 -
5.1.2	<i>set_dhcp</i>	- 149 -
5.1.3	<i>set_dhcp_dns</i>	- 150 -
5.1.4	<i>set_pppoe</i>	- 151 -
5.1.5	<i>set_ip4_local_address</i>	- 152 -
5.1.6	<i>set_ip4_subnet_mask</i>	- 153 -
5.1.7	<i>set_ip4_gateway_address</i>	- 154 -
5.1.8	<i>set_ip4_dns_address</i>	- 155 -
5.1.9	<i>set_pppoe_id</i>	- 156 -
5.1.10	<i>set_pppoe_pwd</i>	- 157 -
5.1.11	<i>set_ip6</i>	- 157 -
5.1.12	<i>set_ip6_gua_type</i>	- 158 -
5.1.13	<i>set_ip6_eui_type</i>	- 159 -
5.1.14	<i>set_ip6_local_address</i>	- 160 -

5.1.15	<i>set_ip6_subnet_prefix_length</i>	- 161 -
5.1.16	<i>set_ip6_gateway_address</i>	- 162 -
5.1.17	<i>set_ip6_dns_address</i>	- 163 -
5.2	Serial Port	- 164 -
5.2.1	<i>set_uart_serial_type</i>	- 164 -
5.2.2	<i>set_uart_ttl</i>	- 165 -
5.2.3	<i>set_uart_baudrate</i>	- 166 -
5.2.4	<i>set_uart_parity</i>	- 167 -
5.2.5	<i>set_uart_databit</i>	- 168 -
5.2.6	<i>set_uart_stopbit</i>	- 169 -
5.2.7	<i>set_uart_flow_control</i>	- 170 -
5.2.8	<i>set_uart_dtrdsr</i>	- 171 -
5.2.9	<i>set_uart_tx_delay</i>	- 172 -
5.3	TCP/IP	- 173 -
5.3.1	<i>set_uart_communication_mode</i>	- 173 -
5.3.2	<i>set_uart_peer_address</i>	- 175 -
5.3.3	<i>set_uart_peer_port</i>	- 176 -
5.3.4	<i>set_uart_local_port</i>	- 177 -
5.3.5	<i>set_uart_cod_tcp_server</i>	- 178 -
5.3.6	<i>set_uart_watermark</i>	- 179 -
5.3.7	<i>set_uart_timeout</i>	- 180 -
5.3.8	<i>set_uart_data_frame_interval</i>	- 181 -
5.3.9	<i>set_uart_separator_length</i>	- 182 -
5.3.10	<i>set_uart_separator_type</i>	- 183 -
5.3.11	<i>set_uart_separator</i>	- 184 -
5.3.12	<i>set_uart_tcp_nodelay</i>	- 186 -
5.3.13	<i>set_uart_rfc2217</i>	- 187 -
5.3.14	<i>set_uart_protocol</i>	- 188 -
5.4	CSC-HR2	- 189 -
5.4.1	<i>set_csc_hr2_communication_mode</i>	- 189 -
5.4.2	<i>set_csc_hr2_id</i>	- 190 -
5.4.3	<i>set_csc_hr2_network_timeout</i>	- 191 -
5.4.4	<i>set_csc_hr2_network_threshold</i>	- 192 -
5.4.5	<i>set_csc_hr2_server_timeout</i>	- 193 -
5.4.6	<i>set_csc_hr2_server_threshold</i>	- 194 -
5.4.7	<i>set_csc_hr2_checkport</i>	- 195 -
5.4.8	<i>set_csc_hr2_first_server_address</i>	- 195 -
5.4.9	<i>set_csc_hr2_first_server_port</i>	- 196 -
5.4.10	<i>set_csc_hr2_second_server_address</i>	- 197 -
5.4.11	<i>set_csc_hr2_second_server_port</i>	- 198 -
5.5	I/O Controller	- 199 -
5.5.1	<i>set_io_http</i>	- 199 -
5.5.2	<i>set_io_http_port</i>	- 200 -
5.5.3	<i>set_io_html_size</i>	- 201 -
5.5.4	<i>set_io_modbus</i>	- 202 -
5.5.5	<i>set_io_input_notification</i>	- 203 -
5.5.6	<i>set_io_output_automatic_initialize</i>	- 204 -
5.5.7	<i>set_io_modbus_type</i>	- 205 -
5.5.8	<i>set_io_unit_id</i>	- 206 -
5.5.9	<i>set_io_input_address</i>	- 207 -
5.5.10	<i>set_io_output_address</i>	- 208 -
5.5.11	<i>set_io_poll_interval</i>	- 209 -
5.5.12	<i>set_io_slave_control_type</i>	- 210 -
5.5.13	<i>set_io_master_control_type</i>	- 211 -
5.5.14	<i>set_io_tcp_type</i>	- 212 -
5.5.15	<i>set_io_multi_connection_number</i>	- 213 -
5.5.16	<i>set_io_modbus_peer_address</i>	- 214 -

5.5.17	<i>set_io_modbus_peer_port</i>	- 215 -
5.5.18	<i>set_io_modbus_local_port</i>	- 216 -
5.5.19	<i>set_io_event_notification</i>	- 216 -
5.5.20	<i>set_event_notification_email</i>	- 218 -
5.5.21	<i>set_io_event_notification_port</i>	- 219 -
5.5.22	<i>set_io_input_valid_time</i>	- 220 -
5.5.23	<i>set_io_macro</i>	- 221 -
5.5.24	<i>set_io_port_macro</i>	- 222 -
5.5.25	<i>set_io_macro_text</i>	- 223 -
5.5.26	<i>set_io_output_delay</i>	- 224 -
5.5.27	<i>set_io_output_initial_state</i>	- 225 -
5.5.28	<i>set_io_comment</i>	- 226 -
5.6	Wireless Network	- 227 -
5.6.1	<i>set_wlan_type</i>	- 227 -
5.6.2	<i>set_wlan_channel</i>	- 228 -
5.6.3	<i>set_wlan_ssid</i>	- 229 -
5.6.4	<i>set_wlan_antenna</i>	- 230 -
5.6.5	<i>set_wlan_phy_mode</i>	- 231 -
5.6.6	<i>set_wlan_short_preamble</i>	- 232 -
5.6.7	<i>set_wlan_short_slot</i>	- 234 -
5.6.8	<i>set_wlan_cts_protection</i>	- 235 -
5.6.9	<i>set_wlan_background_scan</i>	- 236 -
5.6.10	<i>set_wlan_encryption_type</i>	- 237 -
5.6.11	<i>set_wlan_authentication_type</i>	- 238 -
5.6.12	<i>set_wlan_wep_key_length</i>	- 240 -
5.6.13	<i>set_wlan_wep_key_index</i>	- 241 -
5.6.14	<i>set_wlan_wep_key_data_type</i>	- 242 -
5.6.15	<i>set_wlan_wep_key</i>	- 243 -
5.6.16	<i>set_wlan_wpa_passphrase</i>	- 244 -
5.6.17	<i>set_wlan_shared_key</i>	- 245 -
5.6.18	<i>set_wlan_wpa_enterprise</i>	- 246 -
5.6.19	<i>set_wlan_wpa_enterprise_id</i>	- 247 -
5.6.20	<i>set_wlan_wpa_enterprise_pwd</i>	- 248 -
5.7	Options.....	- 249 -
5.7.1	<i>set_telnet</i>	- 249 -
5.7.2	<i>set_send_mac_address</i>	- 250 -
5.7.3	<i>set_ssl</i>	- 251 -
5.7.4	<i>set_ssh</i>	- 252 -
5.7.5	<i>set_ip4_address_search</i>	- 253 -
5.7.6	<i>set_remote_debug</i>	- 254 -
5.7.7	<i>set_tcp_multi_connection</i>	- 255 -
5.7.8	<i>set_power_management</i>	- 256 -
5.7.9	<i>set_comment</i>	- 257 -
5.7.10	<i>set_allowed_mac_address</i>	- 258 -
5.7.11	<i>set_allowed_ip4_address</i>	- 259 -
5.7.12	<i>set_allowed_ip4_network_mask_type</i>	- 260 -
5.7.13	<i>set_allowed_ezmanager</i>	- 261 -
5.7.14	<i>set_allowed_ip6_address</i>	- 262 -
5.7.15	<i>set_allowed_ip6_subnet_prefix_length</i>	- 262 -
5.7.16	<i>set_ip4_change_notification_type</i>	- 263 -
5.7.17	<i>set_ip4_change_notification_data_type</i>	- 264 -
5.7.18	<i>set_ip4_change_notification_interval</i>	- 265 -
5.7.19	<i>set_ip4_change_notification_peer_port</i>	- 266 -
5.7.20	<i>set_ip4_change_notification_peer_address</i>	- 267 -
5.7.21	<i>set_ip4_change_notification_ddns_id</i>	- 268 -
5.7.22	<i>set_ip4_change_notification_ddns_pwd</i>	- 269 -
6	WRITE PARAMETERS	- 271 -

6.1	EzTCP_Write.....	- 271 -
6.2	EzTCP_Initialize.....	- 273 -
7	MANAGEMENT PASSWORD	- 276 -
7.1	EzTCP_ChangePassword	- 276 -
8	STATUS MONITORING	- 280 -
8.1	EzTCP_Status.....	- 280 -
8.2	get_status_string_length	- 281 -
8.3	get_status_string	- 281 -
8.4	get_tcpip4_session_count	- 283 -
8.5	get_tcpip6_session_count	- 284 -
8.6	get_tcpip4_session_info.....	- 286 -
8.7	get_tcpip6_session_info.....	- 288 -
8.8	EzTCP_CloseTcpIp4	- 289 -
8.9	EzTCP_CloseTcpIp6	- 292 -
9	CERTIFICATES.....	- 295 -
9.1	EzTCP_ReadCert.....	- 295 -
9.2	get_certificates_string_length	- 296 -
9.3	get_certificates_string	- 296 -
9.4	EzTCP_WriteSelfSignedCertificates	- 298 -
9.5	EzTCP_WriteCertificates	- 299 -
10	ADDITIONAL FUNCTIONS	- 302 -
10.1	get_version.....	- 302 -
10.2	is_ip6	- 303 -
10.3	is_uart_rs232.....	- 303 -
10.4	is_uart_rs485.....	- 304 -
10.5	is_uart_rs422.....	- 304 -
10.6	is_uart_ttl.....	- 305 -
10.7	is_uart_8_databit_only.....	- 306 -
10.8	is_uart_7_and_8_databit_only	- 306 -
10.9	is_uart_one5_stopbit.....	- 307 -
10.10	is_uart_dtrdsr.....	- 307 -
10.11	is_uart_xonxoff.....	- 308 -
10.12	is_uart_tx_delay.....	- 309 -
10.13	is_uart_data_frame_interval	- 309 -
10.14	is_uart_separator.....	- 310 -
10.15	is_uart_tcp_nodelay	- 310 -
10.16	is_uart_rfc2217	- 311 -
10.17	is_uart_protocol	- 312 -
10.18	is_uart_tcp_server.....	- 312 -
10.19	is_uart_at_command.....	- 313 -
10.20	is_uart_tcp_client.....	- 314 -
10.21	is_uart_udp	- 314 -
10.22	is_uart_serial_modbus	- 315 -
10.23	is_csc_hr2.....	- 316 -
10.24	is_io	- 316 -
10.25	is_io_output automatic_initialize	- 317 -
10.26	is_event_notification.....	- 317 -
10.27	is_wlan.....	- 318 -
10.28	is_wlan_soft_ap	- 319 -
10.29	is_wlan_antenna	- 319 -
10.30	is_wlan_phy_mode.....	- 320 -
10.31	is_wlan_background_scan	- 320 -
10.32	is_wlan_rsn.....	- 321 -

10.33	is_wlan_authentication_type.....	- 322 -
10.34	is_wlan_wpa_enterprise.....	- 322 -
10.35	is_send_mac_address.....	- 323 -
10.36	is_ssl	- 323 -
10.37	is_ssh	- 324 -
10.38	is_remote_debug.....	- 325 -
10.39	is_tcp_multi_connection.....	- 325 -
10.40	is_power_management	- 326 -
10.41	is_password.....	- 326 -
11	EXEMPTION FROM LIABILITY.....	- 328 -
11.1	Exemption from Liability	- 328 -
11.1.1	<i>English</i>	- 328 -
11.1.2	<i>French</i>	- 328 -
12	HISTORY	- 330 -

1 Overview

1.1 Overview

- The main functions provided by ezManager, a product (ezTCP) setting program, are provided in the form of a library. Therefore, you can make programs like ezManager with this library.
- ezManager Library is provided as a form of static and dynamic-link library.

1.2 Conventions Used in This Document

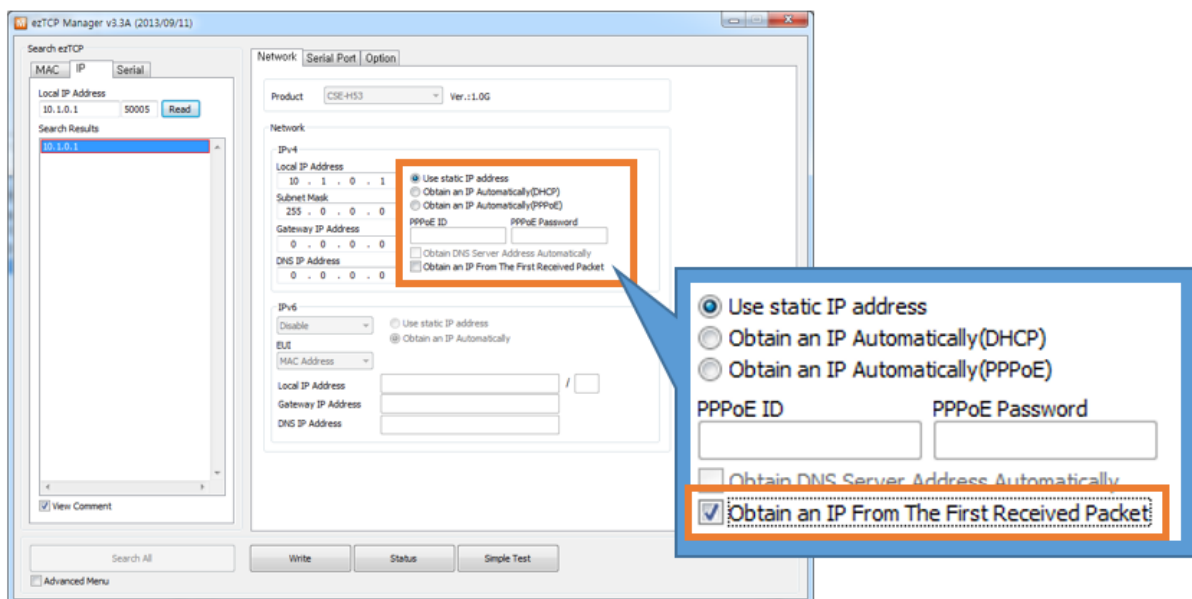
- Throughout this document, a bold text started with '[' and ended with ']' designates the text on the ezManger program.

4.1.1 get_arp

```
int get_arp(unsigned char *mac_address);
```

Description

get_arp retrieves whether **[Obtain an IP From The First Received Packet]** is selected or not.



1.3 WARNING

- We DO NOT guarantee any damage occurred by illegal use of this library.

2 Library initialization and finalization

2.1 Start_Library

```
void Start_Library(int max_unit_number = 256);
```

Description

Start_Library performs dynamic memory allocation to use ezManager libraries and initialize variables used by library functions.

Parameters

int max_unit_number

Set the maximum number of products (ezTCP) that can be searched using EzTCP_Search. If this value is not specified, the default is 256.

Return values

Examples

See also

End_Library

Remarks

Start_Library should be called as soon as an application program starts in order to safely use library functions.

2.2 End_Library

```
void End_Library(void);
```

Description

End_Library releases the use of dynamically allocated memory by a previous call to *Start_Library*.

Parameters

Return values

Examples



See also

Start_Library

Remarks

End_Library should be called to release the dynamically allocated memory when user's program exits. Otherwise memory leak may occur.

3 Search ezTCP

3.1 EzTCP_Search

```
int EzTCP_Search(int udp_port_number, int *error);
```

Description

EzTCP_Search finds ezTCP on the local network. It sends an encrypted UDP packet and waits a response from ezTCP for 2 seconds. The basic UDP port number is 50005 and 50007.

Parameters

int udp_port_number

To designate a UDP port number for *EzTCP_Search*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function execution fails, an error code will be saved.

Return values

If no error occurs, this function returns *EZTCP_SUCCESS*. Otherwise it returns *EZTCP_ERR* and saves a specific error code to *error*.

Examples

```
int error = 0;
int res = 0;

Start_Library();

res = EzTCP_Search(0, &error);

if ( res == EZTCP_ERR )
{
    LPVOID lpMsgBuf;

    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, error, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpMsgBuf, 0, NULL);

    printf("Error[%d] %s\r\n", error, (char*)lpMsgBuf);
}
else
{
    int i = 0;

    int eztcp_count = get_eztcp_count();
```

```

    for(i = 0; i < eztcp_count; i++)
    {
        struct eztcp_info out_eztcp_info;
        int ret = get_eztcp_info_by_index(i, &out_eztcp_info);

        if(ret == EZTCP_SUCCESS)
        {
            printf("MAC Addr.[%02X: %02X: %02X: %02X: %02X: %02X]\r\n",
                out_eztcp_info.mac_address[0],out_eztcp_info.mac_address[1],
                out_eztcp_info.mac_address[2],out_eztcp_info.mac_address[3],
                out_eztcp_info.mac_address[4],out_eztcp_info.mac_address[5]);
        }
    }
}

End_Library();

```

See also

EzTCP_Read, get_eztcp_count, get_eztcp_info_by_index

Remarks

EzTCP_Search uses UDP broadcast packets. Note that these packets may be blocked by firewall or security programs on user's computer system.

3.2 EzTCP_Read

```
int EzTCP_Read(int search_method, unsigned char *mac_address, char *ip_address,
               int udp_port_number, int *error);
```

Description

EzTCP_Read finds a specific ezTCP on the network with a MAC or IP address

Parameters

int search_method

Enter 0 to search ezTCP with a MAC address and enter 1 to search ezTCP with IP address.

*unsigned char *mac_address*

MAC address of ezTCP (6 bytes).

*char *ip_addr*



IP address of ezTCP.

int udp_port_number

To designate a UDP port number for *EzTCP_Read*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

If no error occurs, this function returns *EZTCP_SUCCESS*. Otherwise it returns *EZTCP_ERR* and saves a specific error code to *error*.

Examples

```
int res = 0;
int error = 0;
unsigned char mac_address[6];
char *ip_addr = "10.1.0.1";

mac_address[0] = 0x00;
mac_address[1] = 0x30;
mac_address[2] = 0xf9;
mac_address[3] = 0x00;
mac_address[4] = 0x00;
mac_address[5] = 0x01;

Start_Library();

// MAC address
res = EzTCP_Read(0, mac_address, NULL, 0, &error);
if ( res == EZTCP_SUCCESS )
{
    int i = 0;
    int eztcp_count = get_eztcp_count();

    for(i = 0; i < eztcp_count; i++)
    {
        struct eztcp_info out_eztcp_info;
        int ret = get_eztcp_info_by_index(i, &out_eztcp_info);

        if(ret == EZTCP_SUCCESS)
        {
            printf("MAC Addr.[%02x: %02x: %02x: %02x: %02x: %02x]\r\n",
```

```

        out_eztcp_info.mac_address[0],out_eztcp_info.mac_address[1],
        out_eztcp_info.mac_address[2],out_eztcp_info.mac_address[3],
        out_eztcp_info.mac_address[4],out_eztcp_info.mac_address[5]);
    }
}

// IP address
res = EzTCP_Read(1, NULL, ip_addr, 0, &error);
if ( res == EZTCP_SUCCESS )
{
    int i = 0;
    int eztcp_count = get_eztcp_count();

    for(i = 0; i < eztcp_count; i++)
    {
        struct eztcp_info out_eztcp_info;
        int ret = get_eztcp_info_by_index(i, &out_eztcp_info);

        if(ret == EZTCP_SUCCESS)
        {
            printf("IP Addr. [%s]\r\n", out_eztcp_info.hostname);
        }
    }
}

End_Library();

```

See also

EzTCP_Search, get_eztcp_count, get_eztcp_info_by_index

Remarks

EzTCP_Read may use UDP broadcast packets to search ezTCP on the network depends on the value of *search_method*. Note that these packets may be blocked by firewall or security programs on user's computer.

3.3 get_eztcp_count

```
int get_eztcp_count(void);
```

Description

get_eztcp_count shows the total number of the searched ezTCP on the network by *EzTCP_Search* or *EzTCP_Read*.

Parameters

Return values

The total number of the searched ezTCP on the network.

Examples

```
int error = 0;
int res = 0;

Start_Library();

res = EzTCP_Search(0, &error);

if ( res == EZTCP_ERR )
{
    LPVOID lpMsgBuf;

    FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, error, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpMsgBuf, 0, NULL);

    printf("Error[%d] %s\r\n", error, (char*)lpMsgBuf);
}
else
{
    int i = 0;
    int eztcp_count = get_eztcp_count();

    for(i = 0; i < eztcp_count; i++)
    {
        struct eztcp_info out_eztcp_info;
        int ret = get_eztcp_info_by_index(i, &out_eztcp_info);

        if(ret == EZTCP_SUCCESS)
        {
            printf("MAC Addr. [%02X: %02X: %02X: %02X: %02X: %02X]\r\n",
out_eztcp_info.mac_address[0], out_eztcp_info.mac_address[1],
out_eztcp_info.mac_address[2], out_eztcp_info.mac_address[3],
out_eztcp_info.mac_address[4], out_eztcp_info.mac_address[5]);
        }
    }
}
```



```

}

End_Library();

```

See also

EzTCP_Search, EzTCP_Read, get_eztcp_info_by_index

Remarks

3.4 get_eztcp_info_by_index

```
int get_eztcp_info_by_index(int index, struct eztcp_info *out_eztcp_info);
```

Description

get_eztcp_info_by_index refers to information about the searched ezTCP.

Parameters

int index

The ordinal number of the searched ezTCP. It starts from 0.

*struct eztcp_info *out_eztcp_info*

A structure pointer of *eztcp_info* to save information of the searched ezTCP.

Return values

If the value of *index* is valid, it returns *EZTCP_SUCCESS* and saves information of ezTCP to *out_eztcp_info*. Otherwise it returns *EZTCP_ERR*.

Examples

```

int error = 0;
int res = 0;

start_Library();

res = EzTCP_Search(0, &error);

if ( res == EZTCP_ERR )
{
    LPVOID lpMsgBuf;

```

```

        FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, error, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpMsgBuf, 0, NULL);

        printf("Error[%d] %s\r\n", error, (char*)lpMsgBuf);
    }
else
{
    int i = 0;
    int eztcp_count = get_eztcp_count();

    for(i = 0; i < eztcp_count; i++)
    {
        struct eztcp_info out_eztcp_info;
        int ret = get_eztcp_info_by_index(i, &out_eztcp_info);

        if(ret == EZTCP_SUCCESS)
        {
            printf("MAC Addr. [%02X: %02X: %02X: %02X: %02X: %02X]\r\n",
out_eztcp_info.mac_address[0],out_eztcp_info.mac_address[1],
out_eztcp_info.mac_address[2],out_eztcp_info.mac_address[3],
out_eztcp_info.mac_address[4],out_eztcp_info.mac_address[5]);
        }
    }
}

End_Library();

```

See also

EzTCP_Search, EzTCP_Read, get_eztcp_count

Remarks

eztcp_info structure.

```

struct eztcp_info
{
    unsigned char    mac_address[6];
    char            hostname[NI_MAXHOST];
    unsigned char    comment[65];
    int             env_status;
    char            product_name[32];
    int             search_method;
    int             search_port;
}

```

}

mac_address

MAC address of ezTCP.

hostname

IP address of ezTCP. *NI_MAXHOST* is 1025, as defined by *ezManagerLibrary.h*.

comment

Comment of ezTCP.

env_status

Status of environmental variables. If it is valid, 1 will be shown. If it is invalid, 0 will be shown.

product_name

Product name of ezTCP.

search_method

Searching method for found ezTCP. If *search_method* is 0 then MAC address is used. If *search_method* is 1 then IP address is used.

search_port

UDP port number for found ezTCP.

4 Read parameters

4.1 Network

4.1.1 get_arp

```
int get_arp(unsigned char *mac_address);
```

Description

get_arp retrieves whether [**Obtain an IP From The First Received Packet**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_arp returns 1 if [**Obtain an IP From The First Received Packet**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

4.1.2 get_dhcp

```
int get_dhcp(unsigned char *mac_address);
```

Description

get_dhcp retrieves whether [**Obtain an IP Automatically(DHCP)**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_dhcp returns 1 if [**Obtain an IP Automatically(DHCP)**] is selected, otherwise it returns 0.



If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_dhcp_dns`

Remarks

4.1.3 `get_dhcp_dns`

```
int get_dhcp_dns(unsigned char *mac_address);
```

Description

get_dhcp_dns retrieves whether [**Obtain DNS Server Address Automatically**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_dhcp_dns returns 1 if [**Obtain DNS Server Address Automatically**] is selected..Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_dhcp`

Remarks

ezTCP uses a DNS server address received from a DHCP server when [**Obtain DNS Server Address Automatically**] is selected, otherwise it uses the [**DNS IP Address**] parameter.

4.1.4 `get_pppoe`

```
int get_pppoe(unsigned char *mac_address);
```

Description



get_pppoe retrieves whether [**Obtain an IP Automatically(PPPoE)**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_pppoe returns 1 if [**Obtain an IP Automatically(PPPoE)**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_pppoe_id, *get_pppoe_pwd*

Remarks

4.1.5 get_ip4_local_address

```
int get_ip4_local_address(unsigned char *mac_address, char *out_address);
```

Description

get_ip4_local_address retrieves IPv4 [**Local IP Address**] of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that IPv4 [**Local IP Address**] is saved.

Return values

If no error occurs, *get_ip4_local_address* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);
```

```

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    memset(buf, 0x00, 128);
    get_ip4_local_address(out_eztcp_info.mac_address, buf);
    printf("Local IPv4 : %s\r\n", buf);
}

```

See also

`get_ip4_local_address`, `get_ip4_subnet_mask`, `get_ip4_gateway_address`, `get_ip4_dns_address`

Remarks

get_ip4_local_address saves IPv4 [**Local IP Address**] to *out_address* in a null-terminated IPv4 dotted-decimal notation string.

4.1.6 get_ip4_subnet_mask

```
int get_ip4_subnet_mask(unsigned char *mac_address, char *out_subnet_mask);
```

Description

get_ip4_subnet_mask retrieves IPv4 [**Subnet Mask**] of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_subnet_mask*

A pointer to char that IPv4 [**Subnet Mask**] is saved.

Return values

If no error occurs, *get_ip4_subnet_mask* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];

```

```

    memset(buf, 0x00, 128);
    get_ip4_subnet_mask(out_eztcp_info.mac_address, buf);
    printf("Subnet mask : %s\r\n", buf);
}

```

See also

get_ip4_local_address, get_ip4_subnet_mask, get_ip4_gateway_address, get_ip4_dns_address

Remarks

get_ip4_subnet_mask saves IPv4 [Subnet Mask] to *out_subnet_mask* in a null-terminated IPv4 dotted-decimal notation string.

4.1.7 get_ip4_gateway_address

```
int get_ip4_gateway_address(unsigned char *mac_address, char *out_address);
```

Description

get_ip4_gateway_address retrieves IPv4 [Gateway IP Address] of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that IPv4 [Gateway IP Address] is saved.

Return values

If no error occurs, *get_ip4_gateway_address* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    memset(buf, 0x00, 128);
    get_ip4_gateway_address(out_eztcp_info.mac_address, buf);
    printf("Gateway : %s\r\n", buf);
}

```



```
}`
```

See also

`get_ip4_local_address`, `get_ip4_subnet_mask`, `get_ip4_gateway_address`, `get_ip4_dns_address`

Remarks

`get_ip4_gateway_address` saves IPv4 [Gateway IP Address] to `out_address` in a null-terminated IPv4 dotted-decimal notation string.

4.1.8 `get_ip4_dns_address`

```
int get_ip4_dns_address(unsigned char *mac_address, char *out_address);
```

Description

`get_ip4_dns_address` retrieves IPv4 [DNS IP Address] of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that IPv4 [DNS IP Address] is saved.

Return values

If no error occurs, `get_ip4_dns_address` returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    memset(buf, 0x00, 128);
    get_ip4_dns_address(out_eztcp_info.mac_address, buf);
    printf("DNS : %s\r\n", buf);
}
```

See also



`get_ip4_local_address`, `get_ip4_subnet_mask`, `get_ip4_gateway_address`, `get_ip4_dns_address`.

Remarks

`get_ip4_dns_address` saves IPv4 [DNS IP Address] to `out_address` in a null-terminated IPv4 dotted-decimal notation string.

4.1.9 `get_pppoe_id`

```
int get_pppoe_id(unsigned char *mac_address, char *out_pppoe_id);
```

Description

`get_pppoe_id` retrieves [PPPoE ID] of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_pppoe_id*

A pointer to char that [PPPoE ID] is saved.

Return values

If no error occurs, `get_pppoe_id` returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    memset(buf, 0x00, 128);
    get_ppoe_id(out_eztcp_info.mac_address, buf);
    printf("PPPoE ID : %s\r\n", buf);
}
```

See also

`get_pppoe`, `get_pppoe_pwd`

Remarks



The length of [PPPoE ID] is 32 bytes except a NULL byte. So, *out_pppoe_id* should have at least 33 bytes memory space.

4.1.10 get_pppoe_pwd

```
int get_pppoe_pwd(unsigned char *mac_address, char *out_pppoe_pwd);
```

Description

get_pppoe_pwd retrieves [PPPoE Password] of ezTCP.

*get_pppoe_pwd*는 제품(ezTCP)의 [PPPoE 비밀번호]를 *out_pppoe_pwd*에 저장합니다.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_pppoe_pwd*

A pointer to char that [PPPoE Password] is saved.

Return values

If no error occurs, *get_pppoe_pwd* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    memset(buf, 0x00, 128);
    get_ppoe_pwd(out_eztcp_info.mac_address, buf);
    printf("PPPoE password : %s\r\n", buf);
}
```

See also

get_pppoe, *get_pppoe_id*

Remarks

The length of [PPPoE Password] is 16 bytes except a NULL byte. So, *out_pppoe_pwd* should have at least 17 bytes memory space.

4.1.11 get_ip6

```
int get_ip6(unsigned char *mac_address);
```

Description

get_ip6 retrieves whether **IPv6** is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ip6 returns 1 if **IPv6** is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_ip6, *get_ip6_gua_type*, *get_ip6_eui_type*, *get_ip6_local_address*, *get_ip6_subnet_prefix_length*, *get_ip6_gateway_address*, *get_ip6_dns_address*.

Remarks

4.1.12 get_ip6_gua_type

```
int get_ip6_gua_type(unsigned char *mac_address);
```

Description

get_ip6_gua_type retrieves type of IPv6 [**Local IP Address**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ip6_gua_type returns type of IPv6 [**Local IP Address**]. Each value is defined as follows:

0

[Obtain an IP Automatically]

I

[Use static IP address]

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_ip6(out_eztcp_info.mac_address) == 1)
    {
        int gua_type = get_ip6_gua_type(out_eztcp_info.mac_address);
        if(gua_type == 0)
            printf("[IPv6] Network Type : Obtain an IP automatically.\r\n");
        else if(gua_type == 1)
            printf("[IPv6] Network Type : Use static IP address.\r\n");
    }
}
```

See also

`is_ip6`, `get_ip6`, `get_ip6_eui_type`, `get_ip6_local_address`, `get_ip6_subnet_prefix_length`,
`get_ip6_gateway_address`, `get_ip6_dns_address`

Remarks

4.1.13 get_ip6_eui_type

```
int get_ip6_eui_type(unsigned char *mac_address);
```

Description

get_ip6_eui_type retrieves a method of making Interface ID of IPv6 address.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ip6_eui_type returns a method of making Interface ID of IPv6 address. Each value is defined as



follows:

0

[MAC Address]

1

[Random]

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_ip6(out_eztcp_info.mac_address) == 1)
    {
        int eui_type = get_ip6_eui_type(out_eztcp_info.mac_address);
        if(gua_type == 0)
            printf("[IPv6] EUI : MAC address.\r\n");
        else if(gua_type == 1)
            printf("[IPv6] EUI : Random.\r\n");
    }
}
```

See also

is_ip6, get_ip6, get_ip6_gua_type, get_ip6_local_address, get_ip6_subnet_prefix_length,
get_ip6_gateway_address, get_ip6_dns_address

Remarks

4.1.14 get_ip6_local_address

```
int get_ip6_local_address(unsigned char *mac_address, char *out_address);
```

Description

get_ip6_local_address retrieves IPv6 [Local IP Address] of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



*char *out_address*

A pointer to char that IPv6 [Local IP Address] is saved.

Return values

If no error occurs, *get_ip6_local_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp.mac_address;
    if(is_ip6(mac_address) == 1)
    {
        char buf[128];
        memset(buf, 0x00, 128);
        get_ip6_local_address(mac_address, buf);
        printf("Local IPv6 : %s\r\n", buf);
    }
}
```

See also

is_ip6, *get_ip6*, *get_ip6_gua_type*, *get_ip6_eui_type*, *get_ip6_local_address_prefix*,
get_ip6_gateway_address, *get_ip6_dns_address*

Remarks

4.1.15 get_ip6_subnet_prefix_length

```
int get_ip6_subnet_prefix_length(unsigned char *mac_address,
                                char *out_subnet_prefix_length);
```

Description

get_ip6_subnet_prefix_length retrieves IPv6 subnet prefix length of ezTCP.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_subnet_prefix_length*

A pointer to char that IPv6 subnet prefix length is saved.

Return values

If no error occurs, *get_ip6_subnet_prefix_length* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp.mac_address;
    if(is_ip6(mac_address) == 1)
    {
        char buf[128];
        memset(buf, 0x00, 128);
        get_ip6_subnet_prefix_length(mac_address, buf);
        printf("Subnet prefix length : %s\r\n", buf);
    }
}
```

See also

is_ip6, *get_ip6*, *get_ip6_gua_type*, *get_ip6_eui_type*, *get_ip6_local_address*, *get_ip6_gateway_address*, *get_ip6_dns_address*

Remarks

4.1.16 get_ip6_gateway_address

```
int get_ip6_gateway_address(unsigned char *mac_address, char *out_address);
```

Description

get_ip6_gateway_address retrieves IPv6 [Gateway IP Address] of ezTCP.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that IPv6 [Gateway IP Address] is saved.

Return values

If no error occurs, *get_ip6_gateway_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp.mac_address;
    if(is_ip6(mac_address) == 1)
    {
        char buf[128];
        memset(buf, 0x00, 128);
        get_ip6_gateway_address(mac_address, buf);
        printf("Gateway IPv6 : %s\r\n", buf);
    }
}
```

See also

is_ip6, *get_ip6*, *get_ip6_gua_type*, *get_ip6_eui_type*, *get_ip6_local_address*, *get_ip6_subnet_prefix_length*, *get_ip6_dns_address*

Remarks

4.1.17 get_ip6_dns_address

```
int get_ip6_dns_address(unsigned char *mac_address, char *out_address);
```

Description

get_ip6_dns_address retrieves IPv6 [DNS IP Address] of ezTCP.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that IPv6 [DNS IP Address] is saved.

Return values

If no error occurs, *get_ip6_dns_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_ip6(out_eztcp_info.mac_address) == 1)
    {
        char buf[128];
        memset(buf, 0x00, 128);
        get_ip6_dns_address(out_eztcp_info.mac_address, buf);
        printf("DNS IPV6 : %s\r\n", buf);
    }
}
```

See also

is_ip6, *get_ip6*, *get_ip6_gua_type*, *get_ip6_eui_type*, *get_ip6_local_address*, *get_ip6_subnet_prefix_length*, *get_ip6_gateway_address*.

Remarks

4.2 Serial Port

4.2.1 get_uart_count

```
int get_uart_count(unsigned char *mac_address);
```

Description

get_uart_count retrieves the number of serial ports on ezTCP.

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_uart_count returns the number of serial ports on the ezTCP.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_min_baudrate, *get_uart_max_baudrate*, *get_uart_baudrate*, *get_uart_ttl*, *get_uart_telnet_type*,
get_uart_tx_delay, *get_uart_parity*, *get_uart_databit*, *get_uart_stopbit*, *get_uart_flow_control*,
get_uart_dtrdsr.

Remarks

4.2.2 *get_uart_min_baudrate*

```
int get_uart_min_baudrate(unsigned char *mac_address);
```

Description

get_uart_min_baudrate retrieves the minimum baudrate of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_uart_min_baudrate returns the minimum baudrate of ezTCP.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_count, *get_uart_max_baudrate*, *get_uart_baudrate*, *get_uart_serial_type*, *get_uart_ttl*,
get_uart_tx_delay, *get_uart_parity*, *get_uart_databit*, *get_uart_stopbit*, *get_uart_flow_control*,
get_uart_dtrdsr.

Remarks



4.2.3 get_uart_max_baudrate

```
int get_uart_max_baudrate(unsigned char *mac_address, int uart_index);
```

Description

get_uart_max_baudrate retrieves the maximum baudrate of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_max_baudrate returns the maximum baudrate of ezTCP.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_count, *get_uart_min_baudrate*, *get_uart_baudrate*, *get_uart_serial_type*, *get_uart_ttl*,
get_uart_tx_delay, *get_uart_parity*, *get_uart_databit*, *get_uart_stopbit*, *get_uart_flow_control*,
get_uart_dtrdsr.

Remarks

The maximum baudrate of ezTCP may be changed by another environmental variables.

4.2.4 get_uart_serial_type

```
int get_uart_serial_type(unsigned char *mac_address, int uart_index);
```

Description

get_uart_serial_type retrieves [Serial Type] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_serial_type returns [Serial Type] of a COM port designated by *uart_index*. Each value is defined as follows:

0

RS-232

1

RS-485

2

RS-422

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *get_uart_count*, *get_uart_min_baudrate*,
get_uart_max_baudrate, *get_uart_baudrate*, *get_uart_ttl*, *get_uart_tx_delay*, *get_uart_parity*,
get_uart_databit, *get_uart_stopbit*, *get_uart_flow_control*, *get_uart_dtrdsr*.

Remarks**4.2.5 get_uart_ttl**

```
int get_uart_ttl(unsigned char *mac_address, int uart_index);
```

Description

get_uart_ttl retrieves whether [TTL] of a COM port designated by *uart_index* is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_ttl returns 1 if [TTL] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_uart_count`, `get_uart_min_baudrate`, `get_uart_max_baudrate`, `get_uart_baudrate`, `get_uart_serial_type`, `is_uart_ttl`, `get_uart_tx_delay`, `get_uart_parity`, `get_uart_databit`, `get_uart_stopbit`, `get_uart_flow_control`, `get_uart_dtrdsr`

Remarks**4.2.6 get_uart_baudrate**

```
int get_uart_baudrate(unsigned char *mac_address, int uart_index);
```

Description

get_uart_baudrate retrieves [**Baudrate**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_baudrate returns [**Baudrate**] of a COM port designated by *uart_index*.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

`get_uart_count`, `get_uart_min_baudrate`, `get_uart_max_baudrate`, `get_uart_serial_type`, `get_uart_ttl`, `get_uart_tx_delay`, `get_uart_parity`, `get_uart_databit`, `get_uart_stopbit`, `get_uart_flow_control`, `get_uart_dtrdsr`,

Remarks

The baudrate of a COM port designated by *uart_index* may be changed by another environmental variables.

4.2.7 get_uart_parity

```
int get_uart_parity(unsigned char *mac_address, int uart_index);
```

Description

get_uart_parity retrieves [**Parity**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_parity returns [**Parity**] of a COM port designated by *uart_index*. Each value is defined as follows:

0

NONE

1

EVEN

2

ODD

3

MARK

4

SPACE

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_count, *get_uart_min_baudrate*, *get_uart_max_baudrate*, *get_uart_baudrate*, *get_uart_ttl*, *get_uart_tx_delay*, *get_uart_databit*, *get_uart_stopbit*, *get_uart_flow_control*, *get_uart_dtrdsr*,

Remarks

4.2.8 get_uart_databit

```
int get_uart_databit(unsigned char *mac_address, int uart_index);
```

Description

get_uart_databit retrieves [**Data Bits**] of a COM port designated by *uart_index*.

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_databit returns [Data Bits] of a COM port designated by *uart_index*. Each value is defined as follows:

0

5-bit

1

6-bit

2

7-bit

3

8-bit

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_count, *get_uart_min_baudrate*, *get_uart_max_baudrate*, *get_uart_baudrate*, *is_uart_ttl*, *get_uart_tx_delay*, *get_uart_parity*, *is_uart_8_databit_only*, *is_uart_7_and_8_databit_only*, *get_uart_stopbit*, *get_uart_flow_control*, *get_uart_dtrdsr*

Remarks

if *is_uart_8_databit_only* returns 1 then a ezTCP supports only 8-bit.

if *is_uart_7_and_8_databit_only* return 1 then a ezTCP supports only 7 and 8-bit.

4.2.9 get_uart_stopbit

```
int get_uart_stopbit(unsigned char *mac_address, int uart_index);
```

Description

get_uart_stopbit retrieves [Stop Bit] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_stopbit returns [**Stop Bit**] of a COM port designated by *uart_index*. Each value is defined as follows:

0

1-bit

1

1.5-bit

2

2-bit

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_count, *get_uart_min_baudrate*, *get_uart_max_baudrate*, *get_uart_baudrate*, *get_uart_ttl*,
get_uart_tx_delay, *get_uart_parity*, *get_uart_databit*, *is_uart_one5_stopbit*, *get_uart_flow_control*,
get_uart_dtrdsr,

Remarks

In case of *is_uart_one5_stopbit* returns 1, a ezTCP can use 1.5-bit.

4.2.10 get_uart_flow_control

```
int get_uart_flow_control(unsigned char *mac_address, int uart_index);
```

Description

get_uart_flow_control retrieves [**Flow Control**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_flow_control returns [**Flow Control**] of a COM port designated by *uart_index*. Each value

is defined as follows:

- 0
NONE
- 1
RTS/CTS
- 2
XON/XOFF

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_uart_count`, `get_uart_min_baudrate`, `get_uart_max_baudrate`, `get_uart_baudrate`, `get_uart_ttl`,
`get_uart_tx_delay`, `get_uart_parity`, `get_uart_databit`, `get_uart_stopbit`, `get_uart_dtrdsr`.

Remarks

4.2.11 `get_uart_dtrdsr`

```
int get_uart_dtrdsr(unsigned char *mac_address, int uart_index);
```

Description

get_uart_dtrdsr retrieves whether [DTR/DSR] of a COM port designated by *uart_index* is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_dtrdsr returns 1 if [DTR/DSR] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_uart_count`, `get_uart_min_baudrate`, `get_uart_max_baudrate`, `get_uart_baudrate`, `get_uart_ttl`,



get_uart_tx_delay, get_uart_parity, get_uart_databit, get_uart_stopbit, get_uart_flow_control, is_uart_dtrdsr

Remarks

4.2.12 get_uart_tx_delay

```
int get_uart_tx_delay(unsigned char *mac_address, int uart_index);
```

Description

get_uart_tx_delay retrieves [TX Interval] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_tx_delay returns [TX Interval] of a COM port designated by *uart_index*.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_count, get_uart_min_baudrate, get_uart_max_baudrate, get_uart_baudrate, get_uart_ttl, is_uart_tx_delay, get_uart_parity, get_uart_databit, get_uart_stopbit, get_uart_flow_control, get_uart_dtrdsr

Remarks

4.3 TCP/IP

4.3.1 get_uart_communication_mode

```
int get_uart_communication_mode(unsigned char *mac_address, int uart_index);
```

Description

get_uart_communication_mode retrieves [Communication Mode] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_communication_mode returns [**Communication Mode**] of a COM port designated by *uart_index*. Each value is defined as follows:

0

T2S – TCP Server

1

ATC – AT Command

2

COD – TCP Client

3

U2S – UDP

4

Serial Modbus/TCP

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*,
get_uart_peer_address, *get_uart_peer_port*, *get_uart_local_port*, *get_uart_cod_tcp_server*,
get_uart_watermark, *get_uart_timeout*, *get_uart_data_frame_interval*, *get_uart_separator_length*,
get_uart_separator_type, *get_uart_separator*, *get_uart_tcp_nodelay*, *get_uart_rfc2217*, *get_uart_protocol*

Remarks

4.3.2 *get_uart_peer_address*

```
int get_uart_peer_address(unsigned char *mac_address, int uart_index,
                          char *out_address);
```

Description

get_uart_peer_address retrieves [**Peer Address**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *out_address*

A pointer to char that **[Peer Address]** is saved.

Return values

If no error occurs, *get_uart_peer_address* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int i = 0;
    char buf[65];
    int uart_count = get_uart_count(out_eztcp_info.mac_address);
    for(i=0; i<uart_count; i++)
    {
        memset(buf, 0x00, 65);
        get_uart_peer_address(out_eztcp_info.mac_address, i, buf);
        printf("UART[%d] Peer address : %s\r\n", i, buf);
    }
}
```

See also

get_uart_communication_mode, *get_uart_peer_port*, *get_uart_local_port*, *get_uart_cod_tcp_server*, *get_uart_watermark*, *get_uart_timeout*, *get_uart_data_frame_interval*, *get_uart_separator_length*, *get_uart_separator_type*, *get_uart_separator*, *get_uart_tcp_nodelay*, *get_uart_rfc2217*, *get_uart_protocol*

Remarks

The length of **[Peer Address]** is 64 bytes except a NULL byte. So, *out_address* should have at least 65 bytes memory space.

4.3.3 get_uart_peer_port

```
int get_uart_peer_port(unsigned char *mac_address, int uart_index, char *out_port);
```

Description

get_uart_peer_port retrieves [**Peer Port**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *out_port*

A pointer to char that [**Peer Port**] is saved.

Return values

If no error occurs, *get_uart_peer_port* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int i = 0;
    char buf[65];
    int uart_count = get_uart_count(out_eztcp_info.mac_address);
    for(i=0;i<uart_count;i++)
    {
        memset(buf, 0x00, 65);
        get_uart_peer_port(out_eztcp_info.mac_address, i, buf);
        printf("UART[%d] Peer port : %s\r\n", i, buf);
    }
}
```

See also

get_uart_communication_mode, *get_uart_peer_address*, *get_uart_local_port*, *get_uart_cod_tcp_server*, *get_uart_watermark*, *get_uart_timeout*, *get_uart_data_frame_interval*, *get_uart_separator_length*, *get_uart_separator_type*, *get_uart_separator*, *get_uart_tcp_nodelay*, *get_uart_rfc2217*, *get_uart_protocol*

Remarks

4.3.4 get_uart_local_port

```
int get_uart_local_port(unsigned char *mac_address, int uart_index, char *out_port);
```

Description

get_uart_local_port retrieves [**Local Port**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *out_port*

A pointer to char that [**Local Port**] is saved.

Return values

If no error occurs, *get_uart_local_port* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int i = 0;
    char buf[65];
    int uart_count = get_uart_count(out_eztcp_info.mac_address);
    for(i=0; i<uart_count; i++)
    {
        memset(buf, 0x00, 65);
        get_uart_local_port(out_eztcp_info.mac_address, i, buf);
        printf("UART[%d] Local port : %s\r\n", i, buf);
    }
}
```

See also



get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_cod_tcp_server,
get_uart_watermark, get_uart_timeout, get_uart_data_frame_interval, get_uart_separator_length,
get_uart_separator_type, get_uart_separator, get_uart_tcp_nodelay, get_uart_rfc2217, get_uart_protocol

Remarks

4.3.5 get_uart_cod_tcp_server

```
int get_uart_cod_tcp_server(unsigned char *mac_address, int uart_index);
```

Description

get_uart_cod_tcp_server retrieves whether [TCP Server] of a COM port designated by *uart_index* is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_cod_tcp_server returns 1 if [TCP Server] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_local_port,
get_uart_watermark, get_uart_timeout, get_uart_data_frame_interval, get_uart_separator_length,
get_uart_separator_type, get_uart_separator, get_uart_tcp_nodelay, get_uart_rfc2217, get_uart_protocol

Remarks

4.3.6 get_uart_watermark

```
int get_uart_watermark(unsigned char *mac_address, int uart_index, char *out_watermark);
```

Description



get_uart_watermark retrieves [**Event Byte**] or [**Block Size(Byte)**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *out_watermark*

A pointer to char that [**Event Byte**] or [**Block Size(Byte)**] is saved.

Return values

If no error occurs, *get_uart_watermark* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int i = 0;
    char buf[65];
    int uart_count = get_uart_count(out_eztcp_info.mac_address);
    for(i=0;i<uart_count;i++)
    {
        memset(buf, 0x00, 65);
        int comm_mode = get_uart_communication_mode(out_eztcp_info.mac_address, i);
        if(comm_mode==MUX_TYPE_T2S || comm_mode==MUX_TYPE_ATC || comm_mode==MUX_TYPE_COD)
        {
            get_uart_watermark(out_eztcp_info.mac_address, i, buf);
            printf("UART[%d] Event byte : %s\r\n", i, buf);
        }
        else if(comm_mode==MUX_TYPE_U2S)
        {
            get_uart_watermark(out_eztcp_info.mac_address, i, buf);
            printf("UART[%d] Block size (byte) : %s\r\n", i, buf);
        }
        else
        {

```

```

        printf("Not available.\r\n");
    }
}

```

See also

get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_local_port, get_uart_cod_tcp_server, get_uart_timeout, get_uart_data_frame_interval, get_uart_separator_length, get_uart_separator_type, get_uart_separator, get_uart_tcp_nodelay, get_uart_rfc2217, get_uart_protocol

Remarks

[Event Byte] is using when [Communication Mode] is 0(T2S – TCP Server), 1(ATC – AT Command) or 2(COD – TCP Client).

[Block Size(Byte)] is using when [Communication Mode] is 3(U2S – UDP).

4.3.7 get_uart_timeout

```
int get_uart_timeout(unsigned char *mac_address, int uart_index, char *out_timeout);
```

Description

get_uart_timeout retrieves [Timeout] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *out_timeout*

A pointer to char that [Timeout] is saved.

Return values

If no error occurs, *get_uart_timeout* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{

```

```

    int i = 0;
    char buf[65];
    int uart_count = get_uart_count(out_eztcp_info.mac_address);
    for(i=0;i<uart_count;i++)
    {
        memset(buf, 0x00, 65);

        get_uart_timeout(out_eztcp_info.mac_address,i,buf);
        printf("UART[%d] Timeout : %s\r\n", i, buf);
    }
}

```

See also

get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_local_port, get_uart_cod_tcp_server, get_uart_watermark, get_uart_data_frame_interval, get_uart_separator_length, get_uart_separator_type, get_uart_separator, get_uart_tcp_nodelay, get_uart_rfc2217, get_uart_protocol

Remarks

4.3.8 get_uart_data_frame_interval

```

int get_uart_data_frame_interval(unsigned char *mac_address, int uart_index,
                                char *out_data_frame_interval);

```

Description

get_uart_data_frame_interval retrieves [Data Frame Interval(10ms)] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *out_data_frame_interval*

A pointer to char that [Data Frame Interval(10ms)] is saved.

Return values

If no error occurs, *get_uart_data_frame_interval* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int i = 0;
    char buf[65];
    int uart_count = get_uart_count(out_eztcp_info.mac_address);
    unsigned char *mac_address = out_eztcp_info.mac_address;
    for(i=0;i<uart_count;i++)
    {
        memset(buf, 0x00, 65);

        get_uart_data__frame_interval(mac_address, i, buf);
        printf("UART[%d] Data frame interval : %s\r\n", i, buf);
    }
}
```

See also

get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_local_port,
get_uart_cod_tcp_server, get_uart_watermark, get_uart_timeout, is_uart_data_frame_interval,
get_uart_separator_length, get_uart_separator_type, get_uart_separator, get_uart_tcp_nodelay,
get_uart_rfc2217, get_uart_protocol

Remarks

4.3.9 get_uart_separator_length

```
int get_uart_separator_length(unsigned char *mac_address, int uart_index);
```

Description

get_uart_separator_length retrieves [Separator Length] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.



Return values

get_uart_separator_length returns [Separator Length]. The range of [Separator Length] is 0 to 4.
If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

get_uart_communication_mode, *get_uart_peer_address*, *get_uart_peer_port*, *get_uart_local_port*,
get_uart_cod_tcp_server, *get_uart_watermark*, *get_uart_timeout*, *get_uart_data_frame_interval*,
is_uart_separator, *get_uart_separator_type*, *get_uart_separator*, *get_uart_tcp_nodelay*, *get_uart_rfc2217*,
get_uart_protocol

Remarks**4.3.10 get_uart_separator_type**

```
int get_uart_separator_type(unsigned char *mac_address, int uart_index);
```

Description

get_uart_separator_type retrieves [Separator Operation] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_separator_type returns [Separator Operation] of a COM port designated by *uart_index*.
Each value is defined as follows:

0

Transmit Separators

1

Transmit Separators + 1 Byte

2

Transmit Separators + 2 Bytes

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_local_port, get_uart_cod_tcp_server, get_uart_watermark, get_uart_timeout, get_uart_data_frame_interval, is_uart_separator, get_uart_separator_length, get_uart_separator, get_uart_tcp_nodelay, get_uart_rfc2217, get_uart_protocol

Remarks**4.3.11 get_uart_separator**

```
int get_uart_separator(unsigned char *mac_address, int uart_index, int separator_index,
                      char *out_separator);
```

Description

get_uart_separator retrieves [**Separator(HEX)**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int separator_index

The ordinal number of Separators. It starts from 0.

*char *out_separator*

A pointer to char that a separator designated by *separator_index* is saved.

Return values

If no error occurs, *get_uart_separator* returns 1.

If the *separator_index* is less than 0 or greater than 4 then *get_uart_separator* returns -2.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int i = 0;
    int j = 0;
```

```

char buf[65];
int uart_count = get_uart_count(out_eztcp_info.mac_address);
unsigned char *mac_address = out_eztcp_info.mac_address;
for(i=0;i<uart_count;i++)
{
    memset(buf, 0x00, 65);

    int sep_count = get_uart_separator_length(mac_address, i);
    for(j=0;j<sep_count;j++)
    {
        get_uart_separator(mac_address, i, j, buf);
        printf("UART[%d] Separator[%d] - %s\r\n", i, j, buf);
    }
}
}

```

See also

get_uart_communication_mode, get_uart_peer_address, get_uart_peer_port, get_uart_local_port,
get_uart_cod_tcp_server, get_uart_watermark, get_uart_timeout, get_uart_data_frame_interval,
is_uart_separator, get_uart_separator_length, get_uart_separator_type, get_uart_tcp_nodelay,
get_uart_rfc2217, get_uart_protocol

Remarks

4.3.12 get_uart_tcp_nodelay

```
int get_uart_tcp_nodelay(unsigned char *mac_address, int uart_index);
```

Description

get_uart_tcp_nodelay retrieves [**Disable TCP Transmission Delay**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_tcp_nodelay returns 1 if [**Disable TCP Transmission Delay**] is selected, otherwise it returns

0. If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_uart_communication_mode,` `get_uart_peer_address,` `get_uart_peer_port,` `get_uart_local_port,`
`get_uart_cod_tcp_server,` `get_uart_watermark,` `get_uart_timeout,` `get_uart_data_frame_interval,`
`is_uart_separator,` `get_uart_separator_type,` `get_uart_separator,` `is_uart_tcp_nodelay,` `get_uart_rfc2217,`
`get_uart_protocol`

Remarks

4.3.13 get_uart_rfc2217

```
int get_uart_rfc2217(unsigned char *mac_address, int uart_index);
```

Description

get_uart_rfc2217 retrieves [Telnet COM Port Control(RFC2217)] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_rfc2217 returns 1 if [Telnet COM Port Control(RFC2217)] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_uart_communication_mode,` `get_uart_peer_address,` `get_uart_peer_port,` `get_uart_local_port,`
`get_uart_cod_tcp_server,` `get_uart_watermark,` `get_uart_timeout,` `get_uart_data_frame_interval,`
`is_uart_separator,` `get_uart_separator_type,` `get_uart_separator,` `get_uart_tcp_nodelay,` `is_uart_rfc2217,`
`get_uart_protocol`

Remarks



4.3.14 get_uart_protocol

```
int get_uart_protocol(unsigned char *mac_address, int uart_index);
```

Description

get_uart_protocol retrieves [**Protocol**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

get_uart_protocol returns [**Protocol**] of a COM port designated by *uart_index*. Each value is defined as follows:

0

TCP

1

TCP + TELNET

2

TCP + SSL

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_uart_communication_mode, *get_uart_peer_address*, *get_uart_peer_port*, *get_uart_local_port*, *get_uart_cod_tcp_server*, *get_uart_watermark*, *get_uart_timeout*, *get_uart_data_frame_interval*, *is_uart_separator*, *get_uart_separator_type*, *get_uart_separator*, *get_uart_tcp_nodelay*, *get_uart_rfc2217*, *is_uart_protocol*.

Remarks

Currently, this function only supports CSE-T16, CSE-T32 and CSE-T48. A supported products may be added in future. Please refer to product's user manual for more detail information.

4.4 CSC-HR2

4.4.1 get_csc_hr2_communication_mode

```
int get_csc_hr2_communication_mode (unsigned char *mac_address);
```

Description

get_csc_hr2_communication_mode retrieves whether [**Communication Mode**] of CSC-HR2 is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_csc_hr2_communication_mode returns [**Communication Mode**] of CSC-HR2. Each value is defined as follows:

0

Automation

1

Change EZU-100 Firmware

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_csc_hr2, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*, *get_csc_hr2_network_threshold*,
get_csc_hr2_server_timeout, *get_csc_hr2_server_threshold*, *get_csc_hr2_check_port*,
get_csc_hr2_first_server_address, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.2 get_csc_hr2_id

```
int get_csc_hr2_id(unsigned char *mac_address, char *out_id);
```

Description

get_csc_hr2_id retrieves [**ID of CSC-HR2**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_addr*

A pointer to char that [ID of CSC-HR2] is saved.

Return values

If no error occurs, *get_csc_hr2_id* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char csc_hr2_id[17];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(csc_hr2_id, 0x00, 17);
        get_csc_hr2_id(out_eztcp_info.mac_address, csc_hr2_id);
        printf("ID of CSC-HR2 : %s\r\n", csc_hr2_id);
    }
}
```

See also

is_csc_hr2,	get_csc_hr2_communication_mode,	get_csc_hr2_network_timeout,
get_csc_hr2_network_threshold,	get_csc_hr2_server_timeout,	get_csc_hr2_server_threshold,
get_csc_hr2_check_port,	get_csc_hr2_first_server_address,	get_csc_hr2_first_server_port,
get_csc_hr2_second_server_address,	get_csc_hr2_second_server_port.	

Remarks

The length of [ID of CSC-HR2] is 16 bytes except a NULL byte. So, *out_id* should have at least 17 bytes memory space.

4.4.3 get_csc_hr2_network_timeout

```
int get_csc_hr2_network_timeout(unsigned char *mac_address, char *out_timeout);
```

Description



get_csc_hr2_network_timeout retrieves [**Network Timeout**] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_timeout*

A pointer to char that [**Network Timeout**] is saved.

Return values

If no error occurs, *get_csc_hr2_network_timeout* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[64];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 64);
        get_csc_hr2_network_timeout(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Network Timeout : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_threshold*,
get_csc_hr2_server_timeout, *get_csc_hr2_server_threshold*, *get_csc_hr2_check_port*,
get_csc_hr2_first_server_address, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.4 get_csc_hr2_network_threshold

```
int get_csc_hr2_network_threshold(unsigned char *mac_address, char *out_threshold);
```

Description



get_csc_hr2_network_threshold retrieves [Network Threshold] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_threshold*

A pointer to char that [Network Threshold] is saved.

Return values

If no error occurs, *get_csc_hr2_network_threshold* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[64];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 64);
        get_csc_hr2_network_threshold(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Network Threshold : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_server_timeout, *get_csc_hr2_server_threshold*, *get_csc_hr2_check_port*,
get_csc_hr2_first_server_address, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.5 get_csc_hr2_server_timeout

```
int get_csc_hr2_server_timeout(unsigned char *mac_address, char *out_timeout);
```

Description



get_csc_hr2_server_timeout retrieves [Server Timeout] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_timeout*

A pointer to char that [Server Timeout] is saved.

Return values

If no error occurs, *get_csc_hr2_server_timeout* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[64];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 64);
        get_csc_hr2_server_timeout(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Server Timeout : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_threshold*, *get_csc_hr2_check_port*,
get_csc_hr2_first_server_address, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.6 get_csc_hr2_server_threshold

```
int get_csc_hr2_server_threshold(unsigned char *mac_address, char *out_threshold);
```

Description



get_csc_hr2_server_threshold retrieves [Server Threshold] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_threshold*

A pointer to char that [Server Threshold] is saved.

Return values

If no error occurs, *get_csc_hr2_server_threshold* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[64];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 64);
        get_csc_hr2_server_threshold(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Server Threshold : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_timeout*, *get_csc_hr2_check_port*,
get_csc_hr2_first_server_address, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.7 get_csc_hr2_check_port

```
int get_csc_hr2_check_port(unsigned char *mac_address, char *out_check_port);
```

Description



get_csc_hr2_check_port retrieves [**Check Port**] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_threshold*

A pointer to char that [**Check Port**] is saved.

Return values

If no error occurs, *get_csc_hr2_check_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[64];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 64);
        get_csc_hr2_check_port(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Check Port : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_timeout*, *get_csc_hr2_server_threshold*,
get_csc_hr2_first_server_address, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.8 get_csc_hr2_first_server_address

```
int get_csc_hr2_first_server_address(unsigned char *mac_address, char *out_address);
```

Description



get_csc_hr2_first_server_address retrieves **[First Server Address]** of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that **[First Server Address]** is saved.

Return values

If no error occurs, *get_csc_hr2_first_server_address* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 65);
        get_csc_hr2_first_server_address(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] First Server Address : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_timeout*, *get_csc_hr2_server_threshold*,
get_csc_hr2_check_port, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

The length of **[First Server Address]** is 64 bytes except a NULL byte. So, *out_address* should have at least 65 bytes memory space.

4.4.9 get_csc_hr2_first_server_port

```
int get_csc_hr2_first_server_port(unsigned char *mac_address, char *out_port);
```

Description

get_csc_hr2_first_server_port retrieves [**Port Number**] of First Server Address that CSC-HR2 will be connected to.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_port*

A pointer to char that [**Port Number**] is saved.

Return values

If no error occurs, *get_csc_hr2_first_server_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 65);
        get_csc_hr2_first_server_port(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] First Server Port Number : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_timeout*, *get_csc_hr2_server_threshold*,
get_csc_hr2_check_port, *get_csc_hr2_first_server_address*, *get_csc_hr2_second_server_address*,
get_csc_hr2_second_server_port.

Remarks

4.4.10 *get_csc_hr2_second_server_address*

```
int get_csc_hr2_second_server_address(unsigned char *mac_address, char *out_address);
```

Description

get_csc_hr2_second_server_address retrieves [Second Server Address] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [Second Server Address] is saved.

Return values

If no error occurs, *get_csc_hr2_second_server_address* returns 1. Otherwise, 0 is returned.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 65);
        get_csc_hr2_second_server_address(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Second Server Address : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_timeout*, *get_csc_hr2_server_threshold*,
get_csc_hr2_check_port, *get_csc_hr2_first_server_port*, *get_csc_hr2_second_server_port*.

Remarks

The length of [Second Server Address] is 64 bytes except a NULL byte. So, *out_address* should have at least 65 bytes memory space.

4.4.11 get_csc_hr2_second_server_port

```
int get_csc_hr2_second_server_port(unsigned char *mac_address, char *out_port);
```

Description

get_csc_hr2_second_server_port retrieves [**Port Number**] of Second Server Address that CSC-HR2 will be connected to.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_port*

A pointer to char that [**Port Number**] is saved.

Return values

If no error occurs, *get_csc_hr2_second_server_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    if(is_csc_hr2(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 65);
        get_csc_hr2_second_server_port(out_eztcp_info.mac_address, buf);
        printf("[CSC-HR2] Second Server Port Number : %s\r\n", buf);
    }
}
```

See also

is_csc_hr2, *get_csc_hr2_communication_mode*, *get_csc_hr2_id*, *get_csc_hr2_network_timeout*,
get_csc_hr2_network_threshold, *get_csc_hr2_server_timeout*, *get_csc_hr2_server_threshold*,
get_csc_hr2_check_port, *get_csc_hr2_first_server_address*, *get_csc_hr2_second_server_address*.

Remarks



4.5 I/O Controller

4.5.1 get_io_http

```
int get_io_http(unsigned char *mac_address);
```

Description

get_io_http retrieves whether [**Web(HTTP)**] of I/O Controller is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_http returns 1 if [**Web(HTTP)**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_io, *get_io_http*, *get_io_html_size*

Remarks

4.5.2 get_io_http_port

```
int get_io_http_port(unsigned char *mac_address, char *out_http_port);
```

Description

get_io_http_port retrieves [**Web(HTTP) port**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_http_port*

A pointer to char that [**Web(HTTP) Port**] is saved.

Return values



If no error occurs, *get_io_http_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 8);
        get_io_http_port(out_eztcp_info.mac_address, buf);
        printf("[I/O] Web(HTTP) Port : %s\r\n", buf);
    }
}
```

See also

is_io, *get_io_http*, *get_io_html_size*

Remarks

4.5.3 get_io_html_size

```
int get_io_html_size(unsigned char *mac_address);
```

Description

get_io_html_size retrieves [Size of Web(HTTP)] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_html_size returns [Size of Web(HTTP)]. Each value is defined as follows:

0

80KB

1



96KB

2

112KB

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        int html_size = get_io_html_size(out_eztcp_info.mac_address);
        printf("[I/O] Size of Web(HTTP) : ");
        switch(html_size)
        {
            case 0:
                printf("80KB\r\n");
                break;
            case 1:
                printf("96KB\r\n");
                break;
            case 2:
                printf("112KB\r\n");
                break;
        }
    }
}
```

See also

is_io, get_io_http, get_io_http_port.

Remarks

4.5.4 get_io_modbus

```
int get_io_modbus(unsigned char *mac_address);
```

Description



get_io_modbus retrieves whether [Modbus/TCP] control option of I/O Controller is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_modbus returns 1 if [Modbus/TCP] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_io, *get_io_modbus*, *get_io_input_notification*, *get_io_output_automatic_initialize*, *get_io_modbus_type*, *get_io_unit_id*, *get_io_input_address*, *get_io_output_address*, *get_io_poll_interval*, *get_io_slave_control_type*, *get_io_master_control_type*, *get_io_tcp_type*, *get_io_multi_connection_number*, *get_io_modbus_peer_address*, *get_io_modbus_peer_port*, *get_io_modbus_local_port*.

Remarks

4.5.5 get_io_input_notification

```
int get_io_input_notification(unsigned char *mac_address);
```

Description

get_io_input_notification retrieves whether [Notify Input Port Change] of I/O Controller is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_input_notification returns 1 if [Notify Input Port Change] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also



is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

4.5.6 get_io_output_automatic_initialize

```
int get_io_output_automatic_initialize(unsigned char *mac_address);
```

Description

get_io_output_automatic_initialize retrieves whether **[Initialize the output port state(The output port will be changed to the[Initial State] when Modbus/TCP is disconnected.)]** of I/O Controller is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_output_automatic_initialize returns 1 if **[Initialize the output port state(The output port will be changed to the[Initial State] when Modbus/TCP is disconnected.)]** is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_io, get_io_modbus, get_io_input_notification, is_io_output_automatic_initialize, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

Please refer to product's user manual for more detail information.

4.5.7 get_io_modbus_type

```
int get_io_modbus_type(unsigned char *mac_address);
```

Description

get_io_modbus_type retrieves [Master/Slave] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_modbus_type returns [Master/Slave]. Each value is defined as follows:

0

Slave

1

Master

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        int Modbus_type = get_io_modbus_type(out_eztcp_info.mac_address);
        switch(Modbus_type)
        {
            case 0:
                printf("[I/O] This is Modbus slave.\r\n");
                break;

            case 1:
                printf("[I/O] This is Modbus master.\r\n");
                break;

        }
    }
}
```

See also

is_io, *get_io_modbus*, *get_io_input_notification*, *get_io_output_automatic_initialize*, *get_io_modbus_type*, *get_io_unit_id*, *get_io_input_address*, *get_io_output_address*, *get_io_poll_interval*, *get_io_slave_control_type*, *get_io_master_control_type*, *get_io_tcp_type*, *get_io_multi_connection_number*,

`get_io_modbus_peer_address`, `get_io_modbus_peer_port`, `get_io_modbus_local_port`.

Remarks

4.5.8 `get_io_unit_id`

```
int get_io_unit_id(unsigned char *mac_address, char *out_unit_id);
```

Description

get_io_unit_id retrieves [Unit ID] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_unit_id*

A pointer to char that [Unit ID] is saved.

Return values

If no error occurs, *get_io_unit_id* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 8);
        get_io_unit_id(out_eztcp_info.mac_address, buf);
        printf("[I/O] Unit ID : %s\r\n", buf);
    }
}
```

See also

`is_io`, `get_io_modbus`, `get_io_input_notification`, `get_io_output_automatic_initialize`, `get_io_modbus_type`,
`get_io_unit_id`, `get_io_input_address`, `get_io_output_address`, `get_io_poll_interval`,

`get_io_slave_control_type`, `get_io_master_control_type`, `get_io_tcp_type`, `get_io_multi_connection_number`, `get_io_modbus_peer_address`, `get_io_modbus_peer_port`, `get_io_modbus_local_port`.

Remarks

4.5.9 `get_io_input_address`

```
int get_io_input_address(unsigned char *mac_address, char *out_address);
```

Description

get_io_input_address retrieves [**Input Port Base Address**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [**Input Port Base Address**] is saved.

Return values

If no error occurs, *get_io_input_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 8);
        get_io_input_address(out_eztcp_info.mac_address, buf);
        printf("[I/O] Input Port Base Address : %s\r\n", buf);
    }
}
```

See also

`is_io`, `get_io_modbus`, `get_io_input_notification`, `get_io_output_automatic_initialize`, `get_io_modbus_type`,

`get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval,`
`get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number,`
`get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.`

Remarks

4.5.10 `get_io_output_address`

```
int get_io_output_address(unsigned char *mac_address, char *out_address);
```

Description

get_io_output_address retrieves [**Output Port Base Address**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [**Output Port Base Address**] is saved.

Return values

If no error occurs, *get_io_output_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 8);
        get_io_output_address(out_eztcp_info.mac_address, buf);
        printf("[I/O] Output Port Base Address : %s\r\n", buf);
    }
}
```

See also



is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

4.5.11 get_io_poll_interval

```
int get_io_poll_interval(unsigned char *mac_address, char *out_poll_interval);
```

Description

get_io_poll_interval retrieves [**Poll Interval**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_poll_interval*

A pointer to char that [**Poll Interval**] is saved.

Return values

If no error occurs, *get_io_poll_interval* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        memset(buf, 0x00, 8);
        get_io_poll_interval(out_eztcp_info.mac_address, buf);
        printf("[I/O] Poll Interval : %s\r\n", buf);
    }
}
```

See also



is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

4.5.12 get_io_slave_control_type

```
int get_io_slave_control_type(unsigned char *mac_address);
```

Description

get_io_slave_control_type retrieves [Control Method of Slave's Output Ports] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_slave_control_type returns [Control Method of Slave's Output Ports]. Each value is defined as follows:

0

FC 16 (Multiple)

1

FC 05 (Single)

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        int ctrl_type = get_io_slave_control_type(out_eztcp_info.mac_address);
        switch(ctrl_type)
        {
            case 0:
                printf("[I/O] FC 16(Multiple).\r\n");
```

```

        break;
    case 1:
        printf("[I/O] FC 05(Single).\r\n");
        break;
    }
}
}

```

See also

is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

Please refer to product's user manual for more detail information.

4.5.13 get_io_master_control_type

```
int get_io_master_control_type(unsigned char *mac_address);
```

Description

get_io_master_control_type retrieves [Control Method of Master's Output Ports] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_master_control_type returns [Control Method of Master's Output Ports]. Each value is defined as follows:

0

AND

1

OR

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

```



```

if(ret == EZTCP_SUCCESS)
{
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        int ctrl_type = get_io_master_control_type(out_eztcp_info.mac_address);
        switch(ctrl_type)
        {
            case 0:
                printf("[I/O] AND.\r\n");
                break;
            case 1:
                printf("[I/O] OR.\r\n");
                break;
        }
    }
}

```

See also

is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

Please refer to product's user manual for more detail information.

4.5.14 get_io_tcp_type

```
int get_io_tcp_type(unsigned char *mac_address);
```

Description

get_io_tcp_type retrieves method of making a Modbus/TCP connection.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_tcp_type returns method of making a Modbus/TCP connection.. Each value is defined as follows:



0

Passive Connection

1

Active Connection

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        int ctrl_type = get_io_tcp_type(out_eztcp_info.mac_address);
        switch(ctrl_type)
        {
            case 0:
                printf("[I/O] Passive Connection.\r\n");
                break;
            case 1:
                printf("[I/O] Active Connection.\r\n");
                break;
        }
    }
}
```

See also

is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

Please refer to product's user manual for more detail information.

4.5.15 get_io_multi_connection_number

```
int get_io_multi_connection_number(unsigned char *mac_address);
```

Description

get_io_multi_connection_number returns the maximum number of Modbus/TCP clients that I/O Controllers can accept at the same time if I/O controller is worked as a Passive.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_multi_connection_number returns the maximum number of Modbus/TCP clients that I/O Controllers can accept at the same time if I/O controller is worked as a Passive. The range is 1 to 8.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_io, *get_io_modbus*, *get_io_input_notification*, *get_io_output_automatic_initialize*, *get_io_modbus_type*, *get_io_unit_id*, *get_io_input_address*, *get_io_output_address*, *get_io_poll_interval*, *get_io_slave_control_type*, *get_io_master_control_type*, *get_io_tcp_type*, *get_io_multi_connection_number*, *get_io_modbus_peer_address*, *get_io_modbus_peer_port*, *get_io_modbus_local_port*.

Remarks

4.5.16 get_io_modbus_peer_address

```
int get_io_modbus_peer_address(unsigned char *mac_address, char *out_address);
```

Description

get_io_modbus_peer_address retrieves [Peer Address] of I/O Controllers if it is worked as a Modbus/TCP client.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [Peer Address] is saved.

Return values

If no error occurs, *get_io_modbus_peer_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.



Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && get_io_tcp_type(mac_address) == 1)
    {
        memset(buf, 0x00, 65);
        get_io_modbus_peer_address(out_eztcp_info.mac_address, buf);
        printf("[I/O] Peer Address : %s\r\n", buf);
    }
}
```

See also

is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

The length of [**Peer Address**] is 64 bytes except a NULL byte. So, *out_address* should have at least 65 bytes memory space.

4.5.17 get_io_modbus_peer_port

```
int get_io_modbus_peer_port(unsigned char *mac_address, char *out_port);
```

Description

get_io_modbus_peer_port retrieves [**Peer Port**] of I/O Controllers if it is worked as a Modbus/TCP client.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_port*

A pointer to char that [**Peer Port**] is saved.

Return values

If no error occurs, *get_io_modbus_peer_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && get_io_tcp_type(mac_address) == 1)
    {
        memset(buf, 0x00, 65);
        get_io_modbus_peer_port(out_eztcp_info.mac_address, buf);
        printf("[I/O] Peer Port : %s\r\n", buf);
    }
}
```

See also

is_io, *get_io_modbus*, *get_io_input_notification*, *get_io_output_automatic_initialize*, *get_io_modbus_type*, *get_io_unit_id*, *get_io_input_address*, *get_io_output_address*, *get_io_poll_interval*, *get_io_slave_control_type*, *get_io_master_control_type*, *get_io_tcp_type*, *get_io_multi_connection_number*, *get_io_modbus_peer_address*, *get_io_modbus_peer_port*, *get_io_modbus_local_port*.

Remarks

4.5.18 get_io_modbus_local_port

```
int get_io_modbus_local_port(unsigned char *mac_address, char *out_port);
```

Description

get_io_modbus_local_port retrieves **[Local Port]** of I/O Controllers if it is worked as a Modbus/TCP server.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_port*

A pointer to char that **[Local Port]** is saved.

Return values

If no error occurs, *get_io_modbus_local_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && get_io_tcp_type(mac_address) == 0)
    {
        memset(buf, 0x00, 65);
        get_io_modbus_local_port(out_eztcp_info.mac_address, buf);
        printf("[I/O] Local Port : %s\r\n", buf);
    }
}
```

See also

is_io, get_io_modbus, get_io_input_notification, get_io_output_automatic_initialize, get_io_modbus_type, get_io_unit_id, get_io_input_address, get_io_output_address, get_io_poll_interval, get_io_slave_control_type, get_io_master_control_type, get_io_tcp_type, get_io_multi_connection_number, get_io_modbus_peer_address, get_io_modbus_peer_port, get_io_modbus_local_port.

Remarks

4.5.19 get_io_output_number

```
int get_io_output_number(unsigned char *mac_address);
```

Description

get_io_output_port_number returns the number of output ports of I/O Controllers.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



Return values

get_io_output_port_number returns the number of output ports of I/O Controllers.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

is_io, *get_io_output_number*, *get_io_input_number*, *is_event_notification*, *get_io_event_notification*, *get_event_notification_email*, *get_io_event_notification_port*, *get_io_input_valid_time*, *get_io_macro_type*, *get_io_macro*, *get_io_port_macro*, *get_io_macro_text*, *get_io_output_delay*, *get_io_output_initial_state*, *get_io_comment*.

Remarks**4.5.20 get_io_input_number**

```
int get_io_input_number(unsigned char *mac_address);
```

Description

get_io_input_port_number returns the number of input ports of I/O Controllers.

Parameters

*unsigned char *mac_address*

MAC add

ress of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_input_port_number returns the number of input ports of I/O Controllers.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

is_io, *get_io_output_number*, *get_io_input_number*, *is_event_notification*, *get_io_event_notification*, *get_event_notification_email*, *get_io_event_notification_port*, *get_io_input_valid_time*, *get_io_macro_type*, *get_io_macro*, *get_io_port_macro*, *get_io_macro_text*, *get_io_output_delay*, *get_io_output_initial_state*, *get_io_comment*.

Remarks

4.5.21 get_io_event_notification

```
int get_io_event_notification(unsigned char *mac_address);
```

Description

get_io_event_notification retrieves whether [**Notify Input or Output Port Change(Email)**] of I/O Controller is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_event_notification returns 1 if [**Notify Input or Output Port Change(Email)**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_io, *get_io_output_number*, *get_io_input_number*, *is_event_notification*, *get_io_event_notification*, *get_event_notification_email*, *get_io_event_notification_port*, *get_io_input_valid_time*, *get_io_macro_type*, *get_io_macro*, *get_io_port_macro*, *get_io_macro_text*, *get_io_output_delay*, *get_io_output_initial_state*, *get_io_comment*.

Remarks

4.5.22 get_event_notification_email

```
int get_event_notification_email(unsigned char *mac_address, char *out_email);
```

Description

get_event_notification_email retrieves [**Email Address**] that is used to send an email if a status of input or output ports has changed.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_email*

A pointer to char that [**Email Address**] is saved.

Return values

If no error occurs, *get_event_notification_email* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && is_event_notification(mac_address) == 1)
    {
        if(get_io_event_notification(mac_address) == 1)
        {
            memset(buf, 0x00, 65);
            get_event_notification_email(out_eztcp_info.mac_address, buf);
            printf("[I/O] Email Address : %s\r\n", buf);
        }
    }
}
```

See also

is_io, get_io_output_number, get_io_input_number, is_event_notification, get_io_event_notification, get_event_notification_email, get_io_event_notification_port, get_io_input_valid_time, get_io_macro_type, get_io_macro, get_io_port_macro, get_io_macro_text, get_io_output_delay, get_io_output_initial_state, get_io_comment.

Remarks

The length of [Email Address] is 64 bytes include a NULL byte. So, *out_email* should have at least 64 bytes memory space.

4.5.23 get_io_event_notification_port

```
int get_io_event_notification_port(unsigned char *mac_address, int port_flag,
                                  int port_index);
```

Description

get_io_event_notification_port retrieves whether an input or output port designated by *port_flag* and *port_index* is selected to use [Notify Input or Output Port Change(Email)] or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_flag

port_flag is defined as follows:

0

Input ports

1

Output ports

int port_index

The ordinal number of input or output ports. It starts from 0.

Return values

get_io_event_notification_port returns 1 if an input or output port designated by *port_flag* and *port_index* is selected to use [**Notify Input or Output Port Change(Email)**], otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx = 0;
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && is_event_notification(mac_address) == 1)
    {
        if(get_io_event_notification(mac_address) == 1)
        {
            int input_number = get_io_input_number(mac_address);
            int output_number = get_io_output_number(mac_address);

            for(idx = 0; idx < input_number; idx++)
            {
                if(get_io_event_notification_port(mac_address, 0, idx) == 1)
                {
                    printf("[I/O] Input port#%d is enabled\r\n", idx);
                }
                else
                {
                    printf("[I/O] Input port#%d is disabled\r\n", idx);
                }
            }
        }
    }
}
```

```

        }
    }

    for(idx = 0; idx < output_number; idx++)
    {
        if(get_io_event_notification_port(mac_address, 1, idx) == 1)
        {
            printf("[I/O] Output port#%d is enabled\r\n", idx);
        }
        else
        {
            printf("[I/O] Output port#%d is disabled\r\n", idx);
        }
    }
}
}
}

```

See also

is_io, get_io_output_number, get_io_input_number, is_event_notification, get_io_event_notification, get_event_notification_email, get_io_event_notification_port, get_io_input_valid_time, get_io_macro_type, get_io_macro, get_io_port_macro, get_io_macro_text, get_io_output_delay, get_io_output_initial_state, get_io_comment.

Remarks

4.5.24 get_io_input_valid_time

```
int get_io_input_valid_time(unsigned char *mac_address, int port_index,
                           char *out_input_valid_time);
```

Description

get_io_input_valid_time retrieves [Valid Time(ms)] of an input port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of input ports. It starts from 0.

*char *out_input_valid_time*

A pointer to char that **[Valid Time(ms)]** of an input port designated by *port_index* is saved.

Return values

If no error occurs, *get_io_input_valid_time* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    int idx = 0;
    int input_number = get_io_input_number(out_eztcp_info.mac_address);
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        for(idx = 0; idx < input_number; idx++)
        {
            memset(buf, 0x00, 8);
            get_io_input_valid_time(out_eztcp_info.mac_address, idx, buf);
            printf("[I/O] Input Port#%d Valid Time(ms) : %s\r\n", idx, buf);
        }
    }
}
```

See also

is_io, *get_io_output_number*, *get_io_input_number*, *is_event_notification*, *get_io_event_notification*, *get_event_notification_email*, *get_io_event_notification_port*, *get_io_input_valid_time*, *get_io_macro_type*, *get_io_macro*, *get_io_port_macro*, *get_io_macro_text*, *get_io_output_delay*, *get_io_output_initial_state*, *get_io_comment*.

Remarks

4.5.25 get_io_macro_type

```
int get_io_macro_type(unsigned char *mac_address);
```

Description

get_io_macro_type retrieves type of **[Macro]** function.



Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_macro_type returns 1 if I/O Controllers can use [**Macro**] function for output ports separately. Otherwise it returns 0.

In case of *get_io_macro_type* returns 0, if you turn on [**Macro**] function then the all output ports of I/O Controller are going to use [**Macro**] function.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_io, *get_io_output_number*, *get_io_input_number*, *is_event_notification*, *get_io_event_notification*, *get_event_notification_email*, *get_io_event_notification_port*, *get_io_input_valid_time*, *get_io_macro_type*, *get_io_macro*, *get_io_port_macro*, *get_io_macro_text*, *get_io_output_delay*, *get_io_output_initial_state*, *get_io_comment*.

Remarks

4.5.26 get_io_macro

```
int get_io_macro(unsigned char *mac_address);
```

Description

get_io_macro retrieves whether [**Macro**] of I/O Controller is selected or not. This function is valid if *get_io_macro_type* returns 0.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_io_macro returns 1 if [**Macro**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);
```

```

if(ret == EZTCP_SUCCESS)
{
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        if(get_io_macro_type(out_eztcp_info.mac_address) == 0)
        {
            if(get_io_macro(out_eztcp_info.mac_address) == 1)
                printf("[I/O] Macro is enabled.\r\n");
            else
                printf("[I/O] Macro is disabled.\r\n");
        }
    }
}

```

See also

is_io, get_io_output_number, get_io_input_number, is_event_notification, get_io_event_notification, get_event_notification_email, get_io_event_notification_port, get_io_input_valid_time, get_io_macro_type, get_io_macro, get_io_port_macro, get_io_macro_text, get_io_output_delay, get_io_output_initial_state, get_io_comment.

Remarks

4.5.27 get_io_port_macro

```
int get_io_port_macro(unsigned char *mac_address, int port_index);
```

Description

get_io_port_macro retrieves whether [**Macro**] of an output port designated by *port_index* is selected or not. This function is valid if *get_io_macro_type* returns 1.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

Return values

get_io_port_macro returns 1 if [**Macro**] of an output port designated by *port_index* is selected, otherwise it returns 0.



If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx = 0;
    int output_number = 0;
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && get_io_macro_type(mac_address) == 1)
    {
        output_number = get_io_output_number(mac_address);
        for(idx = 0; idx < output_number; idx++)
        {
            if(get_io_port_macro(mac_address, idx) == 1)
                printf("[I/O] Output Port#%d Macro is enabled.\r\n", idx);
            else
                printf("[I/O] Output Port#%d Macro is disabled.\r\n", idx);
        }
    }
}
```

See also

`is_io`, `get_io_output_number`, `get_io_input_number`, `is_event_notification`, `get_io_event_notification`, `get_event_notification_email`, `get_io_event_notification_port`, `get_io_input_valid_time`, `get_io_macro_type`, `get_io_macro`, `get_io_port_macro`, `get_io_macro_text`, `get_io_output_delay`, `get_io_output_initial_state`, `get_io_comment`.

Remarks

4.5.28 get_io_macro_text

```
int get_io_macro_text(unsigned char *mac_address, int port_index, char *out_macro_text);
```

Description

get_io_macro_text retrieves Macro text of an output port designated by *port_index*.

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

*char *out_macro_text*

A pointer to char that Macro text of an output port designated by *port_index* is saved.

Return values

If no error occurs, *get_io_macro_text* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx = 0;
    int output_number = 0;
    char buf[IO_SCRIPT_LEN + 1];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1)
    {
        output_number = get_io_output_number(mac_address);
        for(idx = 0; idx < output_number; idx++)
        {
            memset(buf, 0x00, IO_SCRIPT_LEN + 1);
            get_io_macro_text(mac_address, idx, buf);
            printf("[I/O] Output port#%d Macro : %s\r\n", idx, buf);
        }
    }
}
```

See also

is_io, *get_io_output_number*, *get_io_input_number*, *is_event_notification*, *get_io_event_notification*, *get_event_notification_email*, *get_io_event_notification_port*, *get_io_input_valid_time*, *get_io_macro_type*, *get_io_macro*, *get_io_port_macro*, *get_io_macro_text*, *get_io_output_delay*, *get_io_output_initial_state*, *get_io_comment*.

Remarks

The length of Macro text is 32 bytes except a NULL byte. So, *out_macro_text* should have at least 33

bytes memory space. **IO_SCRIPT_LEN** defines the maximum length of Macro text.

4.5.29 get_io_output_delay

```
int get_io_output_delay(unsigned char *mac_address, int port_index,
                        char *out_output_delay);
```

Description

get_io_output_delay retrieves [**Delay(ms)**] of an output port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

*char *out_output_delay*

A pointer to char that [**Delay(ms)**] of an output port designated by *port_index* is saved.

Return values

If no error occurs, *get_io_output_delay* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    int idx = 0;
    int output_number = get_io_output_number(out_eztcp_info.mac_address);
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        for(idx = 0; idx < output_number; idx++)
        {
            memset(buf, 0x00, 8);
            get_io_output_delay(out_eztcp_info.mac_address, idx, buf);
            printf("[I/O] Output Port#%d Delay(ms) : %s\r\n", idx, buf);
        }
    }
}
```

```
}
}
```

See also

is_io, get_io_output_number, get_io_input_number, is_event_notification, get_io_event_notification, get_event_notification_email, get_io_event_notification_port, get_io_input_valid_time, get_io_macro_type, get_io_macro, get_io_port_macro, get_io_macro_text, get_io_output_delay, get_io_output_initial_state, get_io_comment.

Remarks

4.5.30 get_io_output_initial_state

```
int get_io_output_initial_state(unsigned char *mac_address, int port_index);
```

Description

get_io_output_initial_state retrieves **[Initial State]** of an output port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

Return values

get_io_output_initial_state returns **[Initial State]** of an output port designated by *port_index*. Each value is defined as follows:

0

OFF

1

ON

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx = 0;
```

```

    unsigned char *mac_address = out_eztcp_info.mac_address;
    int output_number = get_io_output_number(mac_address);
    if(is_io(out_eztcp_info.mac_address) == 1)
    {
        for(idx = 0;idx < output_number;idx++)
        {
            if(get_io_output_initial_state(mac_address, idx) == 1)
                printf("[I/O] Output Port#%d initial state is ON\r\n");
            else
                printf("[I/O] Output Port#%d initial state is OFF\r\n");
        }
    }
}

```

See also

is_io, get_io_output_number, get_io_input_number, is_event_notification, get_io_event_notification, get_event_notification_email, get_io_event_notification_port, get_io_input_valid_time, get_io_macro_type, get_io_macro, get_io_port_macro, get_io_macro_text, get_io_output_delay, get_io_output_initial_state, get_io_comment.

Remarks

4.5.31 get_io_comment

```

int get_io_comment(unsigned char *mac_address, int port_flag, int port_index,
                  char *out_comment);

```

Description

get_io_comment retrieves comment of an input or output port designated by *port_flag* and *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_flag

port_flag is defined as follows:

0

Input ports

1

Output ports

int port_index



The ordinal number of input or output ports. It starts from 0.

*char *out_comment*

A pointer to char that comment of an input or output port designated by *port_index* is saved.

Return values

If no error occurs, *get_io_comment* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx = 0;
    char buf[IO_COMMENT_LEN + 1];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_io(mac_address) == 1 && is_event_notification(mac_address) == 1)
    {
        if(get_io_event_notification(mac_address) == 1)
        {
            int input_number = get_io_input_number(mac_address);
            int output_number = get_io_output_number(mac_address);

            for(idx = 0; idx < input_number; idx++)
            {
                memset(buf, 0x00, IO_COMMENT_LEN + 1);
                get_io_comment(mac_address, 0, idx, buf);
                printf("[I/O] Input port#%d Comment : %s\r\n", idx, buf);
            }

            for(idx = 0; idx < output_number; idx++)
            {
                memset(buf, 0x00, IO_COMMENT_LEN + 1);
                get_io_comment(mac_address, 1, idx, buf);
                printf("[I/O] Output port#%d Comment : %s\r\n", idx, buf);
            }
        }
    }
}
```

See also

is_io, get_io_output_number, get_io_input_number, is_event_notification, get_io_event_notification, get_event_notification_email, get_io_event_notification_port, get_io_input_valid_time, get_io_macro_type, get_io_macro, get_io_port_macro, get_io_macro_text, get_io_output_delay, get_io_output_initial_state, get_io_comment.

Remarks

The length of comment is 16 bytes except a NULL byte. So, *out_comment* should have at least 17 bytes memory space. **IO_COMMENT_LEN** defines the maximum length of comment.

4.6 Wireless Network

4.6.1 get_wlan_type

```
int get_wlan_type(unsigned char *mac_address);
```

Description

get_wlan_type retrieves **[WLAN Topology]**.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_type returns **[WLAN Topology]**. Each value is defined as follows:

0

Ad-hoc

1

Infrastructure

2

Soft AP

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
```

```

    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1)
    {
        int wlan_type = get_wlan_type(mac_address);
        if(wlan_type == 0)
            printf("[WLAN] WLAN Topology : Ad-hoc\r\n");
        else if(wlan_type == 1)
            printf("[WLAN] WLAN Topology : Infrastructure\r\n");
        else if(is_wlan_soft_ap(mac_address) == 1 && wlan_type == 2)
            printf("[WLAN] WLAN Topology : Soft AP\r\n");
    }
}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, get_wlan_antenna,
 get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection,
 get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type,
 get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type,
 get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase,
 get_wlan_shared_key, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id,
 get_wlan_wpa_enterprise_pwd.

Remarks

4.6.2 get_wlan_channel

```
int get_wlan_channel(unsigned char *mac_address);
```

Description

get_wlan_channel retrieves [Channel].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_channel returns [Channel].

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
```



```

int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1)
    {
        int wlan_type = get_wlan_type(mac_address);
        if(wlan_type == 0 || (is_wlan_soft_ap(mac_address) == 1 && wlan_type == 2))
            printf("[WLAN] Channel : %d\r\n", get_wlan_channel(mac_address));
    }
}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, get_wlan_antenna,
get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection,
get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type,
get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type,
get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase,
get_wlan_shared_key, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id,
get_wlan_wpa_enterprise_pwd.

Remarks

4.6.3 get_wlan_ssid

```
int get_wlan_ssid(unsigned char *mac_address, char *out_ssid);
```

Description

get_wlan_ssid retrieves [SSID].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_ssid*

A pointer to char that [SSID] is saved.

Return values

If no error occurs, *get_wlan_ssid* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[33];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1)
    {
        memset(buf, 0x00, 33);
        get_wlan_ssid(mac_address, buf);
        printf("[WLAN] SSID : %s\r\n", buf);
    }
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, get_wlan_antenna, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd.

Remarks

The length of [SSID] is 32 bytes except a NULL byte. So, *out_ssid* should have at least 33 bytes memory space.

4.6.4 get_wlan_antenna

```
int get_wlan_antenna(unsigned char *mac_address);
```

Description

get_wlan_antenna retrieves [Antenna] option.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values



get_wlan_antenna returns **[Antenna]** option. Each value is defined as follows:

0

Internal Antenna

1

External Antenna

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_antenna(mac_address) == 1)
    {
        if(get_wlan_antenna(mac_address) == 0)
            printf("[WLAN] Internal Antenna.\r\n");
        else
            printf("[WLAN] External Antenna.\r\n");
    }
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna,
get_wlan_antenna, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog,
get_wlan_cts_protection, get_wlan_background_scan, get_wlan_encryption_type,
get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index,
get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length,
get_wlan_wpa_passphrase, get_wlan_shared_key, get_wlan_wpa_enterprise,
get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd.

Remarks

4.6.5 get_wlan_phy_mode

```
int get_wlan_phy_mode(unsigned char *mac_address);
```

Description

get_wlan_phy_mode retrieves **[Phy Mode]** in the wireless LAN **[Advanced Settings]**.



Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_phy_mode returns [Phy Mode] in the wireless LAN [Advanced Settings]. Each value is defined as follows:

1

802.11

2

802.11b

3

802.11b/g

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_phy_mode(mac_address) == 1)
    {
        int phy_mode = get_wlan_phy_mode(mac_address);

        if(phy_mode == 1 )
            printf("[WLAN] Phy mode : 802.11\r\n");
        else if(phy_mode == 2)
            printf("[WLAN] Phy mode : 802.11b\r\n");
        else if(phy_mode == 3)
            printf("[WLAN] Phy mode : 802.11b/g\r\n");
    }
}
```

See also

is_wlan, *is_wlan_soft_ap*, *get_wlan_type*, *get_wlan_channel*, *get_wlan_ssid*, *is_wlan_antenna*,
get_wlan_antenna, *is_wlan_phy_mode*, *get_wlan_phy_mode*, *get_wlan_short_preamble*,
get_wlan_short_slog, *get_wlan_cts_protection*, *get_wlan_background_scan*, *get_wlan_encryption_type*,
get_wlan_authentication_type, *get_wlan_wep_key_length*, *get_wlan_wep_key_index*,
get_wlan_wep_key_data_type, *get_wlan_wep_key*, *get_wlan_max_wpa_passphrase_length*,

get_wlan_wpa_passphrase, get_wlan_shared_key, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd,

Remarks

Please refer to product's user manual for more detail information.

4.6.6 get_wlan_short_preamble

```
int get_wlan_short_preamble(unsigned char *mac_address);
```

Description

get_wlan_short_preamble retrieves whether [Short Preamble] in the wireless LAN [Advanced Settings] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_short_preamble returns 1 if [Short Preamble] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_phy_mode(mac_address) == 1)
    {
        int phy_mode = get_wlan_phy_mode(mac_address);

        if(phy_mode == 2 || phy_mode == 3)
        {
            if(get_wlan_short_preamble(mac_address) == 1)
                printf("[WLAN] Short Preamble is enabled.\r\n");
            else
                printf("[WLAN] Short Preamble is disabled.\r\n");
        }
    }
}
```

```
}
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key.

Remarks

Please refer to product's user manual for more detail information.

4.6.7 get_wlan_short_slot

```
int get_wlan_short_slot(unsigned char *mac_address);
```

Description

get_wlan_short_slot retrieves whether [**Short Slot**] in the wireless LAN [**Advanced Settings**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_short_slot returns 1 if [**Short Slot**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_phy_mode(mac_address) == 1)
    {
        int phy_mode = get_wlan_phy_mode(mac_address);
    }
}
```

```

        if(phy_mode == 3)
        {
            if(get_wlan_short_slot(mac_address) == 1)
                printf("[WLAN] Short Slot is enabled.\r\n");
            else
                printf("[WLAN] Short Slot is disabled.\r\n");
        }
    }
}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key.

Remarks

Please refer to product's user manual for more detail information.

4.6.8 get_wlan_cts_protection

```
int get_wlan_cts_protection(unsigned char *mac_address);
```

Description

get_wlan_cts_protection retrieves whether [**CTS Protection**] in the wireless LAN [**Advanced Settings**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_cts_protection returns 1 if [**CTS Protection**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
```



```

int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_phy_mode(mac_address) == 1)
    {
        int phy_mode = get_wlan_phy_mode(mac_address);

        if(phy_mode == 3)
        {
            if(get_wlan_cts_protection(mac_address) == 1)
                printf("[WLAN] CTS Protection is enabled.\r\n");
            else
                printf("[WLAN] CTS Protection is disabled.\r\n");
        }
    }
}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key.

Remarks

Please refer to product's user manual for more detail information.

4.6.9 get_wlan_background_scan

```
int get_wlan_background_scan(unsigned char *mac_address);
```

Description

get_wlan_background_scan retrieves whether [**Background Scan**] in the wireless LAN [**Advanced Settings**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_background_scan returns 1 if **[Background Scan]** is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_background_scan(mac_address) == 1)
    {
        if(get_wlan_background_scan(mac_address) == 1)
            printf("[WLAN] Background Scan is enabled.\r\n");
        else
            printf("[WLAN] Background Scan is disabled.\r\n");
    }
}
```

See also

is_wlan, *is_wlan_soft_ap*, *get_wlan_type*, *get_wlan_channel*, *get_wlan_ssid*, *is_wlan_antenna*,
get_wlan_antenna, *is_wlan_phy_mode*, *get_wlan_phy_mode*, *get_wlan_short_preamble*,
get_wlan_short_slog, *get_wlan_cts_protection*, *is_wlan_background_scan*, *get_wlan_background_scan*,
get_wlan_encryption_type, *get_wlan_authentication_type*, *get_wlan_wpa_enterprise*,
get_wlan_wpa_enterprise_id, *get_wlan_wpa_enterprise_pwd*, *get_wlan_wep_key_length*,
get_wlan_wep_key_index, *get_wlan_wep_key_data_type*, *get_wlan_wep_key*,
get_wlan_max_wpa_passphrase_length, *get_wlan_wpa_passphrase*, *get_wlan_shared_key*.

Remarks

Please refer to product's user manual for more detail information.

4.6.10 *get_wlan_encryption_type*

```
int get_wlan_encryption_type(unsigned char *mac_address);
```

Description

get_wlan_encryption_type retrieves **[Encryption]** in the wireless LAN **[Security Settings]**.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_encryption_type retrieves [Encryption] in the wireless LAN [Security Settings]. Each value is defined as follows:

0

NONE

1

WEP

2

WPA

If the ezTCP with a *mac_address* is not found, then it returns -1.

get_wlan_encryption_type returns -2 if *is_wlan_rsn* returns 1;

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 0)
    {
        int enc_type = get_wlan_encryption_type(mac_address);
        if(enc_type == 0)
            printf("[WLAN] Encryption : None.\r\n");
        else if(enc_type == 1)
            printf("[WLAN] Encryption : WEP.\r\n");
        else if(enc_type == 2)
            printf("[WLAN] Encryption : WPA.\r\n");
    }
}
```

See also

is_wlan, *is_wlan_soft_ap*, *get_wlan_type*, *get_wlan_channel*, *get_wlan_ssid*, *is_wlan_antenna*,
get_wlan_antenna, *is_wlan_phy_mode*, *get_wlan_phy_mode*, *get_wlan_short_preamble*,
get_wlan_short_slog, *get_wlan_cts_protection*, *is_wlan_background_scan*, *get_wlan_background_scan*,
is_wlan_rsn, *get_wlan_encryption_type*, *get_wlan_authentication_type*, *get_wlan_wpa_enterprise*,
get_wlan_wpa_enterprise_id, *get_wlan_wpa_enterprise_pwd*, *get_wlan_wep_key_length*,

get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key,
get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key.

Remarks

4.6.11 get_wlan_authentication_type

```
int get_wlan_authentication_type(unsigned char *mac_address);
```

Description

get_wlan_authentication_type retrieves [**Authentication**] or [**Authentication / Encryption**] of WPA in the wireless LAN [**Security Settings**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_authentication_type always returns 1 if *get_wlan_encryption_type* returns 0.

1

Open System

get_wlan_authentication_type returns [**Authentication**] of WEP if *get_wlan_encryption_type* returns 1. Each value is defined as follows:

1

Open System

2

Shared Key

3

Auto

get_wlan_authentication_type returns [**Authentication / Encryption**] of WPA if *get_wlan_encryption_type* returns 2. Each value is defined as follows:

0

WPA PSK – TKIP

1

WPA PSK – AES

2

WPA PSK – TKIP/AES

- 3 WPA2 PSK – TKIP
- 4 WPA2 PSK – AES
- 5 WPA2 PSK- TKIP/AES

If the ezTCP with a *mac_address* is not found, then it returns -1.
get_wlan_authentication_type returns -2 if *is_wlan_rsn* returns 1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 0)
    {
        int enc_type = get_wlan_encryption_type(mac_address);
        int auth_type = get_wlan_authentication_type(mac_address);
        if(enc_type == 0)
        {
            printf("[WLAN] Encyption : None.\r\n");
            if(auth_type == 1)
                printf("[WLAN] Authentication : Open System.\r\n");
        }
        else if(enc_type == 1)
        {
            printf("[WLAN] Encyption : WEP.\r\n");
            if(auth_type == 1)
                printf("[WLAN] Authentication : Open System.\r\n");
            else if(auth_type == 2)
                printf("[WLAN] Authentication : Shared key.\r\n");
            else if(auth_type == 3)
                printf("[WLAN] Authentication : Auth.\r\n");
        }
        else if(enc_type == 2)
        {
            printf("[WLAN] Encyption : WPA.\r\n");
            if(auth_type == 0)
```

```

        printf("[WLAN] Authentication : WPA PSK - TKIP.\r\n");
    else if(auth_type == 1)
        printf("[WLAN] Authentication : WPA PSK - AES.\r\n");
    else if(auth_type == 2)
        printf("[WLAN] Authentication : WPA PSK - TKIP/AES.\r\n");
    else if(auth_type == 3)
        printf("[WLAN] Authentication : WPA2 PSK - TKIP.\r\n");
    else if(auth_type == 4)
        printf("[WLAN] Authentication : WPA2 PSK - AES.\r\n");
    else if(auth_type == 5)
        printf("[WLAN] Authentication : WPA2 PSK - TKIP/AES.\r\n");

    }

}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan, is_wlan_rsn, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd, get_wlan_shared_key.

Remarks

4.6.12 get_wlan_wep_key_length

```
int get_wlan_wep_key_length(unsigned char *mac_address);
```

Description

get_wlan_wep_key_length retrieves WEP key length.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_wep_key_length retrieves WEP key length. Each value is defined as follows:

1

64-bit

2

128-bit

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1)
    {
        int wep_type = get_wlan_wep_key_length(mac_address);
        if(wep_type == 1)
            printf("[WLAN] WEP Key : 64Bit.\r\n");
        else if(wep_type == 2)
            printf("[WLAN] WEP Key : 128Bit.\r\n");
    }
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd, get_wlan_shared_key.

Remarks

4.6.13 get_wlan_wep_key_index

```
int get_wlan_wep_key_index(unsigned char *mac_address);
```

Description

get_wlan_wep_key_index retrieves index of WEP key.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_wep_key_index returns index of WEP key.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1)
    {
        int key_index = get_wlan_wep_key_index(mac_address);
        printf("[WLAN] WEP Key index %d is using now.\r\n", key_index);
    }
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna,
get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble,
get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan,
get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length,
get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key,
get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key,
get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd.

Remarks

4.6.14 get_wlan_wep_key_data_type

```
int get_wlan_wep_key_data_type(unsigned char *mac_address, int wep_key_index);
```

Description

get_wlan_wep_key_data_type retrieves data type of WEP key designated by *wep_key_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



int wep_key_index

The ordinal number of WEP key. The range of *wep_key_index* is 0 to 3.

Return values

get_wlan_wep_key_data_type returns data type of WEP key designated by *key_index*. Each value is defined as follows:

0

HEX

1

ASCII

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_wlan, *is_wlan_soft_ap*, *get_wlan_type*, *get_wlan_channel*, *get_wlan_ssid*, *is_wlan_antenna*,
get_wlan_antenna, *is_wlan_phy_mode*, *get_wlan_phy_mode*, *get_wlan_short_preamble*,
get_wlan_short_slog, *get_wlan_cts_protection*, *is_wlan_background_scan*, *get_wlan_background_scan*,
get_wlan_encryption_type, *get_wlan_authentication_type*, *get_wlan_wep_key_length*,
get_wlan_wep_key_index, *get_wlan_wep_key_data_type*, *get_wlan_wep_key*,
get_wlan_max_wpa_passphrase_length, *get_wlan_wpa_passphrase*, *get_wlan_shared_key*,
get_wlan_wpa_enterprise, *get_wlan_wpa_enterprise_id*, *get_wlan_wpa_enterprise_pwd*.

Remarks

4.6.15 get_wlan_wep_key

```
int get_wlan_wep_key(unsigned char *mac_address, char *out_wep_key);
```

Description

get_wlan_wep_key retrieves WEP key.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_wep_key*

A pointer to char that WEP key is saved.

Return values

If no error occurs, *get_wlan_wep_key* returns 1.



If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int wep_type;
    int key_index;
    int key_type;
    char buf[32];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1)
    {
        wep_type = get_wlan_wep_key_length(mac_address);
        if(wep_type == 1)
            printf("[WLAN] WEP Key : 64Bit.\r\n");
        else if(wep_type == 2)
            printf("[WLAN] WEP Key : 128Bit.\r\n");

        key_index = get_wlan_wep_key_index(mac_address);

        key_type = get_wlan_wep_key_data_type(mac_address, key_index);
        if(key_type == 0)
            printf("[WLAN] WEP Key index %d is hex format\r\n", key_index);
        else if(key_type == 1)
            printf("[WLAN] WEP Key index %d is ASCII format\r\n", key_index);

        memset(buf, 0x00, 32);
        get_wlan_wep_key(mac_address, buf);
        printf("[WLAN] WEP Key index %d is %s\r\n", key_index, buf);
    }
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna,
get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble,
get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan,
get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length,
get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key,
get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key,

`get_wlan_wpa_enterprise`, `get_wlan_wpa_enterprise_id`, `get_wlan_wpa_enterprise_pwd`.

Remarks

The maximum length of WEP key is 26 bytes except a NULL byte if WEP key length is 128-bit and data type is HEX. So, *out_wep_key* should have at least 27 bytes memory space.

4.6.16 `get_wlan_max_wpa_passphrase_length`

```
int get_wlan_max_wpa_passphrase_length(unsigned char *mac_address);
```

Description

get_wlan_max_wpa_passphrase_length retrieves the maximum length of WPA passphrase that the ezTCP designated by *mac_address* can save.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_max_wpa_passphrase_length returns the maximum length of WPA passphrase. The length is 32 or 64.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`is_wlan`, `is_wlan_soft_ap`, `get_wlan_type`, `get_wlan_channel`, `get_wlan_ssid`, `is_wlan_antenna`,
`get_wlan_antenna`, `is_wlan_phy_mode`, `get_wlan_phy_mode`, `get_wlan_short_preamble`,
`get_wlan_short_slog`, `get_wlan_cts_protection`, `is_wlan_background_scan`, `get_wlan_background_scan`,
`get_wlan_encryption_type`, `get_wlan_authentication_type`, `get_wlan_wep_key_length`,
`get_wlan_wep_key_index`, `get_wlan_wep_key_data_type`, `get_wlan_wep_key`,
`get_wlan_max_wpa_passphrase_length`, `get_wlan_wpa_passphrase`, `get_wlan_shared_key`,
`get_wlan_wpa_enterprise`, `get_wlan_wpa_enterprise_id`, `get_wlan_wpa_enterprise_pwd`.

Remarks

The return value of *get_wlan_max_wpa_passphrase_length* include a NULL byte.

4.6.17 `get_wlan_wpa_passphrase`

```
int get_wlan_wpa_passphrase(unsigned char *mac_address, char *out_wpa_passphrase);
```


Description

get_wlan_wpa_passphrase retrieves WPA passphrase.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_wpa_passphrase*

A pointer to char that WPA passphrase is saved.

Return values

If no error occurs, *get_wlan_wpa_passphrase* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

get_wlan_wpa_passphrase returns -2 if *is_wlan_rsn* returns 1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    int max_len;
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 0)
    {
        max_len = get_wlan_max_wpa_passphrase_length(mac_address);
        printf("[WLAN] Maximum WPA passphrase length : %d\r\n", max_len);

        memset(buf, 0x00, 65);
        get_wlan_wpa_passphrase(mac_address, buf);
        printf("[WLAN] WPA passphrase : %s.\r\n", buf);
    }
}
```

See also

is_wlan, *is_wlan_soft_ap*, *get_wlan_type*, *get_wlan_channel*, *get_wlan_ssid*, *is_wlan_antenna*,
get_wlan_antenna, *is_wlan_phy_mode*, *get_wlan_phy_mode*, *get_wlan_short_preamble*,
get_wlan_short_slog, *get_wlan_cts_protection*, *is_wlan_background_scan*, *get_wlan_background_scan*,
get_wlan_encryption_type, *get_wlan_authentication_type*, *get_wlan_wep_key_length*,
get_wlan_wep_key_index, *get_wlan_wep_key_data_type*, *get_wlan_wep_key*,
get_wlan_max_wpa_passphrase_length, *is_wlan_rsn*, *get_wlan_wpa_passphrase*, *get_wlan_shared_key*,

`get_wlan_wpa_enterprise`, `get_wlan_wpa_enterprise_id`, `get_wlan_wpa_enterprise_pwd`.

Remarks

The maximum length of WPA passphrase is 64 bytes include a NULL byte. So, *out_wpa_passphrase* should have at least 64 bytes memory space.

4.6.18 get_wlan_shared_key

```
int get_wlan_shared_key(unsigned char *mac_address, char *out_shared_key);
```

Description

get_wlan_shared_key retrieves [Shared Key].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_shared_key*

A pointer to char that [Shared Key] is saved.

Return values

If no error occurs, *get_wlan_shared_key* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

get_wlan_shared_key returns -2 if *is_wlan_rsn* returns 0.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[64];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 1)
    {
        memset(buf, 0x00, 64);
        get_wlan_shared_key(mac_address, buf);
        printf("[WLAN] Security Settings - Shared Key : %s\r\n", buf);
    }
}
```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, is_wlan_rsn, get_wlan_wpa_passphrase, get_wlan_shared_key, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd.

Remarks

The maximum length of **[Shared Key]** is 64 bytes include a NULL byte. So, *out_shared_key* should have at least 64 bytes memory space.

4.6.19 get_wlan_wpa_enterprise

```
int get_wlan_wpa_enterprise(unsigned char *mac_address);
```

Description

get_wlan_wpa_enterprise retrieves **[802.1X]** in the wireless LAN **[Security Settings]**.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_wlan_wpa_enterprise returns **[802.1X]**. Each value is defined as follows:

0

Disable

1

EAP TLS

2

EAP TTLS

3

PEAP

If the ezTCP with a *mac_address* is not found, then it returns -1.

get_wlan_wpa_enterprise returns -2 if *is_wpa_enterprise* returns 0.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);
```

```

if(ret == EZTCP_SUCCESS)
{
    int _802_1x;
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 1)
    {
        if(is_wpa_enterprise(mac_address) == 1)
        {
            _802_1x = get_wlan_wpa_enterprise(mac_address);
            if(_802_1x == 0)
                printf("[WLAN] 802.1X : Disable.\r\n");
            else if(_802_1x == 1)
                printf("[WLAN] 802.1X : EAP TLS.\r\n");
            else if(_802_1x == 2)
                printf("[WLAN] 802.1X : EAP TTLS.\r\n");
            else if(_802_1x == 3)
                printf("[WLAN] 802.1X : PEAP.\r\n");
        }
    }
}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan, is_wlan_rsn, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key, is_wpa_enterprise, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd.

Remarks

4.6.20 get_wlan_wpa_enterprise_id

```
int get_wlan_wpa_enterprise_id(unsigned char *mac_address, char *out_enterprise_id);
```

Description

get_wlan_wpa_enterprise_id retrieves [ID] of 802.1X in the wireless LAN [Security Settings].

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_enterprise_id*

A pointer to char that [ID] of 802.1X is saved.

Return values

If no error occurs, *get_wlan_wpa_enterprise_id* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

get_wlan_wpa_enterprise_id returns -2 if *is_wpa_enterprise* returns 0.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int _802_1x;
    char buf[33];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 1)
    {
        if(is_wpa_enterprise(mac_address) == 1)
        {
            _802_1x = get_wlan_wpa_enterprise(mac_address);
            if(_802_1x > 0)
            {
                memset(buf, 0x00, 33);
                get_wlan_wpa_enterprise_id(mac_address, buf);
                printf("[WLAN] 802.1X ID : %s.\r\n", buf);
            }
        }
    }
}
```

See also

is_wlan, *is_wlan_soft_ap*, *get_wlan_type*, *get_wlan_channel*, *get_wlan_ssid*, *is_wlan_antenna*,
get_wlan_antenna, *is_wlan_phy_mode*, *get_wlan_phy_mode*, *get_wlan_short_preamble*,
get_wlan_short_slog, *get_wlan_cts_protection*, *is_wlan_background_scan*, *get_wlan_background_scan*,
is_wlan_rsn, *get_wlan_encryption_type*, *get_wlan_authentication_type*, *get_wlan_wep_key_length*,
get_wlan_wep_key_index, *get_wlan_wep_key_data_type*, *get_wlan_wep_key*,
get_wlan_max_wpa_passphrase_length, *get_wlan_wpa_passphrase*, *get_wlan_shared_key*,
is_wpa_enterprise, *get_wlan_wpa_enterprise*, *get_wlan_wpa_enterprise_id*, *get_wlan_wpa_enterprise_pwd*.

Remarks

The maximum length of [ID] is 32 bytes except a NULL byte. So, *out_enterprise_id* should have at least 33 bytes memory space.

4.6.21 get_wlan_wpa_enterprise_pwd

```
int get_wlan_wpa_enterprise_pwd(unsigned char *mac_address,
                                char *out_enterprise_pwd);
```

Description

get_wlan_wpa_enterprise_pwd retrieves [Password] of 802.1X in the wireless LAN [Security Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_enterprise_pwd*

A pointer to char that [Password] of 802.1X is saved.

Return values

If no error occurs, *get_wlan_wpa_enterprise_pwd* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

get_wlan_wpa_enterprise_pwd returns -2 if *is_wpa_enterprise* returns 0.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int _802_1x;
    char buf[33];
    unsigned char *mac_address = out_eztcp_info.mac_address;
    if(is_wlan(mac_address) == 1 && is_wlan_rsn(mac_address) == 1)
    {
        if(is_wpa_enterprise(mac_address) == 1)
        {
            _802_1x = get_wlan_wpa_enterprise(mac_address);
            if(_802_1x >= 2)
            {
```

```

        memset(buf, 0x00, 33);
        get_wlan_wpa_enterprise_pwd(mac_address, buf);
        printf("[WLAN] 802.1X Password : %s.\r\n", buf);
    }
}

```

See also

is_wlan, is_wlan_soft_ap, get_wlan_type, get_wlan_channel, get_wlan_ssid, is_wlan_antenna, get_wlan_antenna, is_wlan_phy_mode, get_wlan_phy_mode, get_wlan_short_preamble, get_wlan_short_slog, get_wlan_cts_protection, is_wlan_background_scan, get_wlan_background_scan, is_wlan_rsn, get_wlan_encryption_type, get_wlan_authentication_type, get_wlan_wep_key_length, get_wlan_wep_key_index, get_wlan_wep_key_data_type, get_wlan_wep_key, get_wlan_max_wpa_passphrase_length, get_wlan_wpa_passphrase, get_wlan_shared_key, is_wpa_enterprise, get_wlan_wpa_enterprise, get_wlan_wpa_enterprise_id, get_wlan_wpa_enterprise_pwd.

Remarks

The maximum length of **[Password]** is 32 bytes except a NULL byte. So, *out_enterprise_pwd* should have at least 33 bytes memory space.

4.7 Options

4.7.1 get_telnet

```
int get_telnet(unsigned char *mac_address);
```

Description

get_telnet retrieves whether **[Telnet]** is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_telnet returns 1 if **[Telnet]** is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_telnet, get_send_mac_address, get_ssl, get_ssh, get_ip4_address_search, get_remote_debug,

`get_tcp_multi_connection`, `get_power_management`, `get_comment`

Remarks

4.7.2 `get_send_mac_address`

```
int get_send_mac_address(unsigned char *mac_address);
```

Description

`get_send_mac_address` retrieves whether [Send MAC Address] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

`get_send_mac_address` returns 1 if [Send MAC Address] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

`get_telnet`, `is_send_mac_address`, `get_send_mac_address`, `get_ssl`, `get_ssh`, `get_ip4_address_search`, `get_remote_debug`, `get_tcp_multi_connection`, `get_power_management`, `get_comment`

Remarks

4.7.3 `get_ssl`

```
int get_ssl(unsigned char *mac_address);
```

Description

`get_ssl` retrieves whether [SSL] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ssl returns 1 if [SSL] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

get_telnet, *get_send_mac_address*, *is_ssl*, *get_ssl*, *get_ssh*, *get_ip4_address_search*, *get_remote_debug*,
get_tcp_multi_connection, *get_power_management*, *get_comment*

Remarks**4.7.4 get_ssh**

```
int get_ssh(unsigned char *mac_address);
```

Description

get_ssh retrieves whether [SSH] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ssh returns 1 if [SSH] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

get_telnet, *get_send_mac_address*, *get_ssl*, *is_ssh*, *get_ssh*, *get_ip4_address_search*, *get_remote_debug*,
get_tcp_multi_connection, *get_power_management*, *get_comment*

Remarks**4.7.5 get_ip4_address_search**

```
int get_ip4_address_search(unsigned char *mac_address);
```

Description

get_ip4_address_search retrieves whether [IPv4 Address Search] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ip4_address_search returns 1 if [IPv4 Address Search] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

get_telnet, *get_send_mac_address*, *get_ssl*, *get_ssh*, *get_ip4_address_search*, *get_remote_debug*,
get_tcp_multi_connection, *get_power_management*, *get_comment*

Remarks**4.7.6 get_remote_debug**

```
int get_remote_debug(unsigned char *mac_address);
```

Description

get_remote_debug retrieves whether [Debugging Message] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_remote_debug returns 1 if [Debugging Message] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

get_telnet, *get_send_mac_address*, *get_ssl*, *get_ssh*, *get_ip4_address_search*, *is_remote_debug*,
get_remote_debug, *get_tcp_multi_connection*, *get_power_management*, *get_comment*



Remarks

4.7.7 get_tcp_multi_connection

```
int get_tcp_multi_connection(unsigned char *mac_address);
```

Description

get_tcp_multi_connection retrieves whether [**Multiple Connection**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_tcp_multi_connection returns 1 if [**Multiple Connection**] is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_telnet, *get_send_mac_address*, *get_ssl*, *get_ssh*, *get_ip4_address_search*, *get_remote_debug*, *is_tcp_multi_connection*, *get_tcp_multi_connection*, *get_power_management*, *get_comment*

Remarks

4.7.8 get_power_management

```
int get_power_management(unsigned char *mac_address);
```

Description

get_power_management retrieves whether [**Power Management**] is selected or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_power_management returns 1 if [**Power Management**] is selected, otherwise it returns 0.



get_power_management returns -2 if *is_power_management* returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

get_telnet, *get_send_mac_address*, *get_ssl*, *get_ssh*, *get_ip4_address_search*, *get_remote_debug*,
get_tcp_multi_connection, *is_power_management*, *get_power_management*, *get_comment*

Remarks

4.7.9 get_comment

```
int get_comment(unsigned char *mac_address, char *out_comment);
```

Description

get_comment retrieves [**Comment**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_comment*

A pointer to char that [**Comment**] is saved.

Return values

If no error occurs, *get_comment* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[65];
    memset(buf, 0x00, 65);
    get_comment(out_eztcp_info.mac_address, buf);
    printf("[ezTCP] Comment : %s.\r\n", buf);
}
```

See also

`get_telnet`, `get_send_mac_address`, `get_ssl`, `get_ssh`, `get_ip4_address_search`, `get_remote_debug`,
`get_tcp_multi_connection`, `get_power_management`, `get_comment`

Remarks

The length of **[Comment]** is 64 bytes except a NULL byte. So, *out_comment* should have at least 65 bytes memory space.

4.7.10 get_allowed_ezmanager

```
int get_allowed_ezmanager(unsigned char *mac_address);
```

Description

get_allowed_ezmanager retrieves whether **[Apply To ezManager]** is selected or not. **[Apply To ezManager]** is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_allowed_ezmanager returns 1 if **[Apply To ezManager]** is selected, otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    if(get_allowed_ezmanager(mac_address) == 1)
        printf("[ezTCP Firewall] Apply To ezManager is enabled.\r\n");
    else
        printf("[ezTCP Firewall] Apply To ezManager is disabled.\r\n");
}
```

See also

`get_allowed_ezmanager`,

`get_allowed_mac_address`,

`get_allowed_ip4_address`,



get_allowed_ip4_network_mask_type, get_allowed_ip6_address, get_allowed_ip6_subnet_prefix_length

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

4.7.11 get_allowed_mac_address

```
int get_allowed_mac_address(unsigned char *mac_address, int mac_index,
                           char *out_mac_address);
```

Description

get_allowed_mac_address retrieves [Allowed MAC Address]. [Allowed MAC Address] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int mac_index

The ordinal number of MAC address. The range of *mac_index* is 0 to 5.

*char *out_mac_address*

A pointer to char that partial of MAC address designated by *mac_index* is saved.

Return values

If no error occurs, *get_allowed_mac_address* returns 1.

If the *mac_index* is less than 0 or greater than 5 then *get_allowed_mac_address* returns -2.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx;
    char buf[3];
    unsigned char *mac_address = out_eztcp_info.mac_address;

    printf("[ezTCP Firewall] Allowed MAC Address : ");
    for(idx = 0; idx < 6; idx++)
```

```

    {
        memset(buf, 0x00, 3);
        get_allowed_mac_address(mac_address, idx, buf);
        printf("%s ", buf);
    }
    printf("\r\n");
}

```

See also

get_allowed_ezmanager, get_allowed_mac_address, get_allowed_ip4_address,
get_allowed_ip4_network_mask_type, get_allowed_ip6_address, get_allowed_ip6_subnet_prefix_length

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

4.7.12 get_allowed_ip4_address

```
int get_allowed_ip4_address(unsigned char *mac_address, char *out_address);
```

Description

get_allowed_ip4_address retrieves [IPv4 Address] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [IPv4 Address] is saved.

Return values

If no error occurs, *get_allowed_ip4_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int idx;

```

```

    int networkmask_type;
    char buf[16];
    unsigned char *mac_address = out_eztcp_info.mac_address;

    printf("[ezTCP Firewall] Allowed IP Range\r\n");

    memset(buf, 0x00, 16);
    get_allowed_ip4_address(mac_address, buf);
    printf("IPv4 Address : %s\r\n", buf);

    networkmask_type = get_allowed_ip4_network_mask_type(mac_address);
    switch(networkmask_type)
    {
    case 0:
        printf("Network Mask : 255.255.255.255\r\n");
        break;
    case 1:
        printf("Network Mask : 255.255.255.0\r\n");
        break;
    case 2:
        printf("Network Mask : 255.255.0.0\r\n");
        break;
    case 3:
        printf("Network Mask : 255.0.0.0\r\n");
        break;
    case 4:
        printf("Network Mask : 0.0.0.0\r\n");
        break;
    }
}

```

See also

get_allowed_ezmanager, get_allowed_mac_address, get_allowed_ip4_address,
get_allowed_ip4_network_mask_type, get_allowed_ip6_address, get_allowed_ip6_subnet_prefix_length

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

4.7.13 get_allowed_ip4_network_mask_type

```
int get_allowed_ip4_network_mask_type(unsigned char *mac_address);
```


Description

get_allowed_ip4_network_mask_type retrieves [Network Mask] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_allowed_ip4_network_mask_type returns [Network Mask] in [Allowed IP Range] section. Each value is defined as follows:

0

255.255.255.255

1

255.255.255.0

2

255.255.0.0

3

255.0.0.0

4

0.0.0.0

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int network_mask_type;
    char buf[16];
    unsigned char *mac_address = out_eztcp_info.mac_address;

    printf("[ezTCP Firewall] Allowed IP Range\r\n");

    memset(buf, 0x00, 16);
    get_allowed_ip4_address(mac_address, buf);
    printf("IPv4 Address : %s\r\n", buf);

    network_mask_type = get_allowed_ip4_network_mask_type(mac_address);
```

```

switch(networkmask_type)
{
case 0:
    printf("Network Mask : 255.255.255.255\r\n");
    break;
case 1:
    printf("Network Mask : 255.255.255.0\r\n");
    break;
case 2:
    printf("Network Mask : 255.255.0.0\r\n");
    break;
case 3:
    printf("Network Mask : 255.0.0.0\r\n");
    break;
case 4:
    printf("Network Mask : 0.0.0.0\r\n");
    break;
}
}

```

See also

get_allowed_ezmanager, get_allowed_mac_address, get_allowed_ip4_address,
get_allowed_ip4_network_mask_type, get_allowed_ip6_address, get_allowed_ip6_subnet_prefix_length

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

4.7.14 get_allowed_ip6_address

```
int get_allowed_ip6_address(unsigned char *mac_address, char *out_address);
```

Description

get_allowed_ip6_address retrieves [IPv6 Address] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [IPv6 Address] is saved.



Return values

If no error occurs, *get_allowed_ip6_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    unsigned char *mac_address = out_eztcp_info.mac_address;

    printf("[ezTCP Firewall] Allowed IP Range\r\n");

    memset(buf, 0x00, 128);
    get_allowed_ip6_address(mac_address, buf);
    printf("IPv6 Address : %s\r\n", buf);

    memset(buf, 0x00, 128);
    get_allowed_ip6_subnet_prefix_length(mac_address, buf);
    printf("\tSubnet Prefix Length : %s\r\n", buf);
}
```

See also

get_allowed_ezmanager, get_allowed_mac_address, get_allowed_ip4_address,
get_allowed_ip4_network_mask_type, get_allowed_ip6_address, get_allowed_ip6_subnet_prefix_length

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

4.7.15 get_allowed_ip6_subnet_prefix_length

```
int get_allowed_ip6_subnet_prefix_length(unsigned char *mac_address,
                                         char *out_subnet_prefix_length);
```

Description

get_allowed_ip6_subnet_prefix_length retrieves **subnet prefix length** of [IPv6 Address] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_subnet_prefix_length*

A pointer to char that subnet prefix length of [IPv6 Address] is saved.

Return values

If no error occurs, *get_allowed_ip6_subnet_prefix_length* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[128];
    unsigned char *mac_address = out_eztcp_info.mac_address;

    printf("[ezTCP Firewall] Allowed IP Range\r\n");

    memset(buf, 0x00, 128);
    get_allowed_ip6_address(mac_address, buf);
    printf("IPv6 Address : %s\r\n", buf);

    memset(buf, 0x00, 128);
    get_allowed_ip6_subnet_prefix_length(mac_address, buf);
    printf("\tSubnet Prefix Length : %s\r\n", buf);
}
```

See also

get_allowed_ezmanager, *get_allowed_mac_address*, *get_allowed_ip4_address*,
get_allowed_ip4_network_mask_type, *get_allowed_ip6_address*, *get_allowed_ip6_subnet_prefix_length*

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

4.7.16 get_ip4_change_notification_type

```
int get_ip4_change_notification_type(unsigned char *mac_address);
```

Description

get_ip4_change_notification_type retrieves [Protocol] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ip4_change_notification_type returns [Protocol] in [Notify IPv4 Change] section. Each value is defined as follows:

0

Disable

1

DDNS(dyndns.org)

2

TCP

3

UDP

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    printf("[Notify IPv4 Change] Protocol : ");

    int protocol_type = get_ip4_change_notification_type(mac_address);
    switch(protocol_type)
    {
        case 0:
            printf("Disable\r\n");
            break;
```

```

        case 1:
            printf("DDNS(dyndns.org)\r\n");
            break;

        case 2:
            printf("TCP\r\n");
            break;

        case 3:
            printf("UDP\r\n");
            break;

    }
}

```

See also

get_ip4_change_notification_type,
get_ip4_change_notification_interval,
get_ip4_change_notification_peer_address,
get_ip4_change_notification_ddns_pwd.

get_ip4_change_notification_data_type,
get_ip4_change_notification_peer_port,
get_ip4_change_notification_ddns_id,

Remarks

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

4.7.17 get_ip4_change_notification_data_type

```
int get_ip4_change_notification_data_type(unsigned char *mac_address);
```

Description

get_ip4_change_notification_data_type retrieves [Data Type] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

get_ip4_change_notification_data_type returns [Data Type] in [Notify IPv4 Change] section. Each value is defined as follows:

0

ASCII

1

HEX

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    int data_type;
    int protocol_type = get_ip4_change_notification_type(mac_address);
    if(protocol_type == 2 || protocol_type == 3)
    {
        printf("[Notify IPv4 Change] Data Type : ");
        data_type = get_ip4_change_notification_data_type(mac_address);
        if(data_type == 0)
            printf("ASCII\r\n");
        else if(data_type == 1)
            printf("HEX\r\n");
    }
}

```

See also

get_ip4_change_notification_type,
get_ip4_change_notification_interval,
get_ip4_change_notification_peer_address,
get_ip4_change_notification_ddns_pwd.

get_ip4_change_notification_data_type,
get_ip4_change_notification_peer_port,
get_ip4_change_notification_ddns_id,

Remarks

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

4.7.18 get_ip4_change_notification_interval

```

int get_ip4_change_notification_interval(unsigned char *mac_address, char
*out_interval);

```

Description

get_ip4_change_notification_interval retrieves [Interval] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_interval*

A pointer to char that **[Interval]** is saved.

Return values

If no error occurs, *get_ip4_change_notification_interval* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    char buf[8];
    int protocol_type = get_ip4_change_notification_type(mac_address);
    if(protocol_type == 2 || protocol_type == 3)
    {
        get_ip4_change_notification_interval(mac_address, buf);
        printf("[Notify IPv4 Change] Interval : %s\r\n", buf);
    }
}
```

See also

get_ip4_change_notification_type,
get_ip4_change_notification_interval,
get_ip4_change_notification_peer_address,
get_ip4_change_notification_ddns_pwd.

get_ip4_change_notification_data_type,
get_ip4_change_notification_peer_port,
get_ip4_change_notification_ddns_id,

Remarks

Please refer to product's user manual for more detail information about **[Notify IPv4 Change]**.

4.7.19 *get_ip4_change_notification_peer_port*

```
int get_ip4_change_notification_peer_port(unsigned char *mac_address, char *out_port);
```

Description

get_ip4_change_notification_peer_port retrieves **[Port]** in **[Notify IPv4 Change]** section.



Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_port*

A pointer to char that [Port] is saved.

Return values

If no error occurs, *get_ip4_change_notification_peer_port* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    char buf[8];
    int protocol_type = get_ip4_change_notification_type(mac_address);
    if(protocol_type == 2 || protocol_type == 3)
    {
        get_ip4_change_notification_peer_port(mac_address, buf);
        printf("[Notify IPv4 Change] Port Number : %s\r\n", buf);
    }
}
```

See also

get_ip4_change_notification_type,
get_ip4_change_notification_interval,
get_ip4_change_notification_peer_address,
get_ip4_change_notification_ddns_pwd.

get_ip4_change_notification_data_type,
get_ip4_change_notification_peer_port,
get_ip4_change_notification_ddns_id,

Remarks

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

4.7.20 *get_ip4_change_notification_peer_address*

```
int get_ip4_change_notification_peer_address(unsigned char *mac_address,
                                             char *out_address);
```

Description

get_ip4_change_notification_peer_address retrieves [Host Name(dyndns)] or [Host Name(custom)] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_address*

A pointer to char that [Host Name(dyndns)] or [Host Name(custom)] is saved.

Return values

If no error occurs, *get_ip4_change_notification_peer_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    char buf[64];
    int protocol_type = get_ip4_change_notification_type(mac_address);
    memset(buf, 0x00, 64);
    get_ip4_change_notification_peer_address(mac_address, buf);

    if(protocol == 1)
    {
        printf("[Notify IPv4 Change] Host Name (dyndns) : %s\r\n", buf);
    }
    else if(protocol_type == 2 || protocol_type == 3)
    {
        printf("[Notify IPv4 Change] Host Name (custom) : %s\r\n", buf);
    }
}
```

See also

get_ip4_change_notification_type,
get_ip4_change_notification_interval,
get_ip4_change_notification_peer_address,

get_ip4_change_notification_data_type,
get_ip4_change_notification_peer_port,
get_ip4_change_notification_ddns_id,



get_ip4_change_notification_ddns_pwd.

Remarks

If [**Protocol**] in [**Notify IPv4 Change**] section is 1(DDNS(dyndns.org)) then [**Host Name(custom)**] will be used. If [**Protocol**] in [**Notify IPv4 Change**] section is 2(TCP) or 3(UDP) then [**Host Name(dyndns)**] will be used.

The length of [**Host Name(dyndns)**] or [**Host Name(custom)**] is 64 bytes include a NULL byte. So, *out_address* should have at least 64 bytes memory space.

Please refer to product's user manual for more detail information about [**Notify IPv4 Change**].

4.7.21 get_ip4_change_notification_ddns_id

```
int get_ip4_change_notification_ddns_id(unsigned char *mac_address, char *out_ddns_id);
```

Description

get_ip4_change_notification_ddns_id retrieves [**DDNS ID**] in [**Notify IPv4 Change**] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_ddns_id*

A pointer to char that [**DDNS IS**] is saved.

Return values

If no error occurs, *get_ip4_change_notification_ddns_id* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    char buf[33];
    int protocol_type = get_ip4_change_notification_type(mac_address);
    if(protocol == 1)
    {
```

```

        memset(buf, 0x00, 33);
        get_ip4_change_notification_ddns_id(mac_address, buf);

        printf("[Notify IPv4 Change] DDNS ID : %s\r\n", buf);
    }
}

```

See also

get_ip4_change_notification_type,	get_ip4_change_notification_data_type,
get_ip4_change_notification_interval,	get_ip4_change_notification_peer_port,
get_ip4_change_notification_peer_address,	get_ip4_change_notification_ddns_id,
get_ip4_change_notification_ddns_pwd.	

Remarks

The length of **[DDNS ID]** is 32 bytes except a NULL byte. So, *out_ddns_id* should have at least 33 bytes memory space.

Please refer to product's user manual for more detail information about **[Notify IPv4 Change]**.

4.7.22 get_ip4_change_notification_ddns_pwd

```

int get_ip4_change_notification_ddns_pwd(unsigned char *mac_address,
                                         char *out_ddns_pwd);

```

Description

get_ip4_change_notification_ddns_pwd retrieves **[DDNS Password]** in **[Notify IPv4 Change]** section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_ddns_pwd*

A pointer to char that **[DDNS Password]** is saved.

Return values

If no error occurs, *get_ip4_change_notification_ddns_pwd* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

```

```

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    char buf[17];
    int protocol_type = get_ip4_change_notification_type(mac_address);
    if(protocol == 1)
    {
        memset(buf, 0x00, 17);
        get_ip4_change_notification_ddns_pwd(mac_address, buf);

        printf("[Notify IPv4 Change] DDNS Password : %s\r\n", buf);
    }
}

```

See also

get_ip4_change_notification_type,
get_ip4_change_notification_interval,
get_ip4_change_notification_peer_address,
get_ip4_change_notification_ddns_pwd.

get_ip4_change_notification_data_type,
get_ip4_change_notification_peer_port,
get_ip4_change_notification_ddns_id,

Remarks

The length of [**DDNS Password**] is 16 bytes except a NULL byte. So, *out_ddns_pwd* should have at least 17 bytes memory space.

Please refer to product's user manual for more detail information about [**Notify IPv4 Change**].

5 Change parameters

5.1 Network

5.1.1 set_arp

```
int set_arp(unsigned char *mac_address, int onoff);
```

Description

set_arp changes the parameter of [Obtain an IP From The First Received Packet].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

If no error occurs, *set_arp* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

5.1.2 set_dhcp

```
int set_dhcp(unsigned char *mac_address, int onoff);
```

Description

set_dhcp changes the parameter of [Obtain an IP Automatically(DHCP)].



Parameters*unsigned char *mac_address*MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.*int onoff*

Each value is defined as follows:

0

Disable

1

Enable

Return valuesIf no error occurs, *set_dhcp* returns 1.If the ezTCP with a *mac_address* is not found, then it returns -1.**Examples****See also***set_dhcp_dns*, *set_pppoe***Remarks**

[**Obtain an IP Automatically(DHCP)**] and [**Obtain an IP Automatically(PPPoE)**] can't be enabled at the same time. So, if you enable [**Obtain an IP Automatically(DHCP)**] then [**Obtain an IP Automatically(PPPoE)**] is disabled automatically.

5.1.3 set_dhcp_dns

```
int set_dhcp_dns(unsigned char *mac_address, int onoff);
```

Description*set_dhcp_dns* changes the parameter of [**Obtain DNS Server Address Automatically**].**Parameters***unsigned char *mac_address*MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.*int onoff*

Each value is defined as follows:

0

Disable

1
Enable

Return values

If no error occurs, *set_dhcp_dns* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

set_dhcp

Remarks

If [**Obtain DNS Server Address Automatically**] is enabled then ezTCP uses a DNS server address received from a DHCP server. Otherwise, you may enter a DNS server address.

5.1.4 set_pppoe

```
int set_pppoe(unsigned char *mac_address, int onoff);
```

Description

set_pppoe changes the parameter of [**Obtain an IP Automatically(PPPoE)**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0
Disable
1
Enable

Return values

If no error occurs, *set_pppoe* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples



See also

set_pppoe_id, set_pppoe_pwd, set_dhcp

Remarks

[**Obtain an IP Automatically(PPPoE)**] and [**Obtain an IP Automatically(DHCP)**] can't be enabled at the same time. So, if you enable [**Obtain an IP Automatically(PPPoE)**] then [**Obtain an IP Automatically(DHCP)**] is disabled automatically.

5.1.5 set_ip4_local_address

```
int set_ip4_local_address(unsigned char *mac_address, char *ip4_local_address);
```

Description

set_ip4_local_address changes the parameter of IPv4 [**Local IP Address**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip4_local_address*

ip4_local_address points to a character string containing an IPv4 network address in dotted-decimal format, "ddd.ddd.ddd.ddd", where ddd is a decimal number of up to three digits in the range 0 to 255.

Return values

Each value returned by *set_ip4_local_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

If *ip4_local_address* is unavailable, then it returns -2.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

char *new_ip4_addr = "10.1.0.2";

if(ret == EZTCP_SUCCESS)
{
```

```

int res = set_ip4_local_address(out_eztcp_info.mac_address, new_ip4_addr);
if(res == 1)
    printf("Success.\r\n");
else if(res == -1)
    printf("There is no product information of the requested MAC address\r\n");
else if(res == -2)
    printf("You entered an incorrect IPv4 address.\r\n");
}

```

See also

set_ip4_local_address, set_ip4_subnet_mask, set_ip4_gateway_address, set_ip4_dns_address.

Remarks

Parameter *ip4_local_address* is null-terminated dotted-decimal notation string.

5.1.6 set_ip4_subnet_mask

```
int set_ip4_subnet_mask(unsigned char *mac_address, char *ip4_subnet_mask);
```

Description

set_ip4_subnet_mask changes the parameter of IPv4 [Subnet Mask].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip4_subnet_mask*

ip4_subnet_mask points to a character string containing an subnet mask in dotted-decimal format, "ddd.ddd.ddd.ddd", where ddd is a decimal number of up to three digits in the range 0 to 255.

Return values

Each value returned by *set_ip4_subnet_mask* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

If *ip4_subnet_mask* is unavailable, then it returns -2.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

char *new_subnet_mask = "255.255.255.0";

if(ret == EZTCP_SUCCESS)
{
    int res = set_ip4_subnet_mask(out_eztcp_info.mac_address, new_subnet_mask);
    if(res == 1)
        printf("Success.\r\n");
    else if(res == -1)
        printf("There is no product information of the requested MAC address\r\n");
    else if(res == -2)
        printf("You entered an incorrect subnet mask.\r\n");
}

```

See also

set_ip4_local_address, set_ip4_subnet_mask, set_ip4_gateway_address, set_ip4_dns_address.

Remarks

Parameter *ip4_subnet_mask* is null-terminated dotted-decimal notation string.

5.1.7 set_ip4_gateway_address

```
int set_ip4_gateway_address(unsigned char *mac_address, char *ip4_gateway_addr);
```

Description

set_ip4_gateway_address changes the parameter of IPv4 [**Gateway IP Address**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip4_gateway_addr*

ip4_gateway_addr points to a character string containing an subnet mask in dotted-decimal format, "ddd.ddd.ddd.ddd", where ddd is a decimal number of up to three digits in the range 0 to 255.

Return values

If no error occurs, *set_ip4_gateway_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.



Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

char *new_gateway = "10.1.0.254";

if(ret == EZTCP_SUCCESS)
{
    int res = set_ip4_gateway_address(out_eztcp_info.mac_address, new_gateway);
    if(res == 1)
        printf("Success.\r\n");
    else if(res == -1)
        printf("There is no product information of the requested MAC address\r\n");
}
```

See also

set_ip4_local_address, set_ip4_subnet_mask, set_ip4_gateway_address, set_ip4_dns_address.

Remarks

Parameter *ip4_gateway_addr* is null-terminated dotted-decimal notation string.

5.1.8 set_ip4_dns_address

```
int set_ip4_dns_address(unsigned char *mac_address, char *ip4_dns_addr);
```

Description

set_ip4_dns_address changes the parameter of IPv4 [DNS IP Address].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip4_dns_addr*

ip4_dns_addr points to a character string containing an subnet mask in dotted-decimal format, "ddd.ddd.ddd.ddd", where ddd is a decimal number of up to three digits in the range 0 to 255.

Return values

If no error occurs, *set_ip4_dns_address* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

char *new_dns = "10.1.0.200";

if(ret == EZTCP_SUCCESS)
{
    int res = set_ip4_dns_address(out_eztcp_info.mac_address, new_dns);
    if(res == 1)
        printf("Success.\r\n");
    else if(res == -1)
        printf("There is no product information of the requested MAC address\r\n");
}
```

See also

set_ip4_local_address, set_ip4_subnet_mask, set_ip4_gateway_address, set_ip4_dns_address.

Remarks

Parameter *ip4_dns_addr* is null-terminated dotted-decimal notation string.

5.1.9 set_pppoe_id

```
int set_pppoe_id(unsigned char *mac_address, char *pppoe_id);
```

Description

set_pppoe_id changes the parameter of [PPPoE ID].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *pppoe_id*

pppoe_id points to a character string containing an user name to log in to PPPoE service.

Return values

If no error occurs, *set_pppoe_id* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples



See also

set_pppoe, set_pppoe_pwd

Remarks

The length of [PPPoE ID] is 32 bytes except a NULL byte.

5.1.10 set_pppoe_pwd

```
int set_pppoe_pwd(unsigned char *mac_address, char *pppoe_pwd);
```

Description

set_pppoe_pwd changes the parameter of [PPPoE Password].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *pppoe_pwd*

pppoe_pwd points to a character string containing a password to log in to PPPoE service.

Return values

If no error occurs, *set_pppoe_pwd* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

get_pppoe, get_pppoe_id

Remarks

The length of [PPPoE Password] is 16 bytes except a NULL byte.

5.1.11 set_ip6

```
int set_ip6(unsigned char *mac_address, int onoff);
```

Description

set_ip6 changes the parameter of [IPv6].

Parameters*unsigned char *mac_address*MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.*int onoff*

Each value is defined as follows:

0

Disable

1

Enable

Return valuesEach value returned by *set_ip6* is defined as follows:*1*

If the function succeeds, it returns 1.

*-1*If the ezTCP with a *mac_address* is not found, then it returns -1.*-2*It returns -2 if *is_ip6* returns 0.*-3*It returns -3 if *onoff* is 1 and [SSL] is enabled.**Examples****See also***is_ip6*, *set_ip6_gua_type*, *set_ip6_eui_type*, *set_ip6_local_address*, *set_ip6_subnet_prefix_length*, *set_ip6_gateway_address*, *set_ip6_dns_address*.**Remarks****5.1.12 set_ip6_gua_type**

```
int set_ip6_gua_type(unsigned char *mac_address, int gua_type);
```

Description*set_ip6_gua_type* changes type of IPv6 [Local IP Address].**Parameters***unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int gua_type

Each value is defined as follows:

0

[Obtain an IP Automatically]

1

[Use static IP address]

Return values

Each value returned by *set_ip6_gua_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_ip6* returns 0.

-3

It returns -3 if *gua_type* is invalid.

Examples

See also

is_ip6, *set_ip6_gua_type*, *set_ip6_eui_type*, *set_ip6_local_address*, *set_ip6_subnet_prefix_length*, *set_ip6_gateway_address*, *set_ip6_dns_address*.

Remarks

5.1.13 set_ip6_eui_type

```
int set_ip6_eui_type(unsigned char *mac_address, int eui_type);
```

Description

set_ip6_eui_type changes a method of making Interface ID. This is only available if *set_ip6_gua_type* returns 0(Obtain an IP Automatically).

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int eui_type

Each value is defined as follows:

- 0
[MAC Address]
- 1
[Random]

Return values

Each value returned by *set_ip6_eui_type* is defined as follows:

- 1
If the function succeeds, it returns 1.
- 1
If the ezTCP with a *mac_address* is not found, then it returns -1.
- 2
It returns -2 if *is_ip6* returns 0.
- 3
It returns -3 if *eui_type* is invalid.

Examples**See also**

is_ip6, *set_ip6_gua_type*, *set_ip6_eui_type*, *set_ip6_local_address*, *set_ip6_subnet_prefix_length*, *set_ip6_gateway_address*, *set_ip6_dns_address*.

Remarks5.1.14 *set_ip6_local_address*

```
int set_local_ip6(unsigned char *mac_address, char *ip6_local_address);
```

Description

set_ip6_local_address changes the parameter of [Local IP Address] in [IPv6] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip6_local_address*

ip6_local_address points to a character string containing an IPv6 address.

Return values

Each value returned by *set_ip6_local_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_ip6* returns 0.

-3

It returns -3 if *ip6_local_address* is invalid.

Examples

See also

is_ip6, *set_ip6_gua_type*, *set_ip6_eui_type*, *set_ip6_local_address*, *set_ip6_subnet_prefix_length*, *set_ip6_gateway_address*, *set_ip6_dns_address*.

Remarks

5.1.15 set_ip6_subnet_prefix_length

```
int set_ip6_subnet_prefix_length(unsigned char *mac_address,
                                int ip6_subnet_prefix_length);
```

Description

set_ip6_subnet_prefix_length changes subnet prefix length of IPv6 address in [IPv6] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int ip6_subnet_prefix_length

ip6_subnet_prefix_length is subnet prefix length of IPv6 address and its range is 0 to 128.

Return values

Each value returned by *set_ip6_subnet_prefix_length* is defined as follows:

1

If the function succeeds, it returns 1.

-1



If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_ip6* returns 0.

-3

It returns -3 if *ip6_subnet_prefix_length* is invalid.

Examples

See also

is_ip6, *set_ip6_gua_type*, *set_ip6_eui_type*, *set_ip6_local_address*, *set_ip6_subnet_prefix_length*, *set_ip6_gateway_address*, *set_ip6_dns_address*.

Remarks

5.1.16 set_ip6_gateway_address

```
int set_ip6_gateway_address(unsigned char *mac_address, char *ip6_gateway_address);
```

Description

set_ip6_gateway_address changes the parameter of [Gateway IP Address] in [IPv6] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip6_gateway_address*

ip6_gateway_address points to a character string containing a IPv6 address of gateway.

Return values

Each value returned by *set_ip6_gateway_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_ip6* returns 0.

-3

It returns -3 if *ip6_gateway_address* is invalid.

Examples

See also

`is_ip6`, `set_ip6_gua_type`, `set_ip6_eui_type`, `set_ip6_local_address`, `set_ip6_subnet_prefix_length`, `set_ip6_gateway_address`, `set_ip6_dns_address`.

Remarks

5.1.17 `set_ip6_dns_address`

```
int set_ip6_dns_address(unsigned char *mac_address, char *ip6_dns_address);
```

Description

`set_ip6_dns_address` changes the parameter of [DNS IP Address] in [IPv6] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip6_dns_address*

ip6_dns_address points to a character string containing a IPv6 address of DNS server.

Return values

Each value returned by `set_ip6_dns_address` is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_ip6* returns 0.

-3

It returns -3 if *ip6_dns_address* is invalid.

Examples

See also

`is_ip6`, `set_ip6_gua_type`, `set_ip6_eui_type`, `set_ip6_local_address`, `set_ip6_subnet_prefix_length`, `set_ip6_gateway_address`, `set_ip6_dns_address`.

Remarks



5.2 Serial Port

5.2.1 set_uart_serial_type

```
int set_uart_serial_type(unsigned char *mac_address, int uart_index, int serial_type);
```

Description

set_uart_serial_type changes the parameter of [**Serial Type**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int serial_type

Each value is defined as follows:

<i>0</i>	RS-232
<i>1</i>	RS-485
<i>2</i>	RS-422

Return values

Each value returned by *set_uart_serial_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

In case of **CSE-M53** or **CSE-M53N**, if you want to use **RS-485** or **RS-422** then [**Baudrate**] should be less than **230,400bps**. Otherwise, *set_uart_serial_type* returns -2.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*, *get_uart_min_baudrate*, *get_uart_max_baudrate*, *set_uart_baudrate*, *set_uart_parity*, *set_uart_databit*, *set_uart_stopbit*, *set_uart_flow_control*, *set_uart_dtrdsr*, *set_uart_tx_delay*.

Remarks

If *is_uart_rs232* returns 1 then RS-232 is available.

If *is_uart_rs485* returns 1 then RS-485 is available.

If *is_uart_rs422* returns 1 then RS-422 is available.

5.2.2 set_uart_ttl

```
int set_uart_ttl(unsigned char *mac_address, int uart_index, int onoff);
```

Description

set_uart_ttl changes the parameter of [TTL] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_uart_ttl* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_ttl* return 0.

Examples**See also**

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*, *get_uart_min_baudrate*, *get_uart_max_baudrate*, *set_uart_baudrate*, *set_uart_parity*, *set_uart_databit*, *set_uart_stopbit*, *set_uart_flow_control*, *set_uart_dtrdsr*, *set_uart_tx_delay*.

Remarks

5.2.3 set_uart_baudrate

```
int set_uart_baudrate(unsigned char *mac_address, int uart_index, int baudrate);
```

Description

set_uart_baudrate changes the parameter of [**Baudrate**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int baudrate

It is a baudrate of a COM port designated by *uart_index*.

Return values

Each value returned by *set_uart_baudrate* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

If a COM port designated by *uart_index* is disabled then *set_uart_baudrate* returns -2.

-3

If *baudrate* is invalid then *set_uart_baudrate* return -3.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*,
get_uart_min_baudrate, *get_uart_max_baudrate*, *set_uart_baudrate*, *set_uart_parity*, *set_uart_databit*,
set_uart_stopbit, *set_uart_flow_control*, *set_uart_dtrdsr*, *set_uart_tx_delay*.

Remarks

[**Baudrate**] of a COM port designated by *uart_index* may be changed by another environmental variables.

5.2.4 set_uart_parity

```
int set_uart_parity(unsigned char *mac_address, int uart_index, int parity);
```

Description

set_uart_parity changes the parameter of [**Parity**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int parity

Each value is defined as follows:

<i>0</i>	NONE
<i>1</i>	EVEN
<i>2</i>	ODD
<i>3</i>	MARK
<i>4</i>	SPACE

Return values

Each value returned by *set_uart_parity* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	In case of <i>is_uart_7_and_8_databit_only</i> returns 1, if you want to use NONE for [Parity] then [Data Bits] should be 8-bit . Otherwise, <i>set_uart_parity</i> returns -2.

Examples

See also



is_uart_rs232, is_uart_rs485, is_uart_rs422, set_uart_serial_type, is_uart_ttl, set_uart_ttl,
 get_uart_min_baudrate, get_uart_max_baudrate, set_uart_baudrate, is_uart_8_databit_only,
 is_uart_7_and_8_databit_only, set_uart_parity, set_uart_databit, set_uart_stopbit, set_uart_flow_control,
 set_uart_dtrdsr, set_uart_tx_delay.

Remarks

If *is_uart_7_and_8_databit_only* returns 1 then the product should use **EVEN, ODD, MARK or SPACE** for [Parity] in order to use **7-bit** for [Data Bits].

5.2.5 set_uart_databit

```
int set_uart_databit(unsigned char *mac_address, int uart_index, int databit);
```

Description

set_uart_databit changes the parameter of [Data Bits] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int databit

Each value is defined as follows:

0

5-bit

1

6-bit

2

7-bit

3

8-bit

Return values

Each value returned by *set_uart_databit* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

In case of *is_uart_7_and_8_databit_only* returns 1, if you want to use **7-bit** for [Data Bits] then [Parity] should **NOT** be **NONE**. Otherwise, *set_uart_databit* returns -2.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*,
get_uart_min_baudrate, *get_uart_max_baudrate*, *set_uart_baudrate*, *is_uart_8_databit_only*,
is_uart_7_and_8_databit_only, *set_uart_parity*, *set_uart_databit*, *set_uart_stopbit*, *set_uart_flow_control*,
set_uart_dtrdsr, *set_uart_tx_delay*.

Remarks

If *is_uart_7_and_8_databit_only* returns 1 then the product should use **EVEN**, **ODD**, **MARK** or **SPACE** for [Parity] in order to use **7-bit** for [Data Bits].

5.2.6 set_uart_stopbit

```
int set_uart_stopbit(unsigned char *mac_address, int uart_index, int stopbit);
```

Description

set_uart_stopbit changes the parameter of [Stop Bit] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int stopbit

Each value is defined as follows:

<i>0</i>	1-bit
<i>1</i>	1.5-bit
<i>2</i>	2-bit

Return values

Each value returned by *set_uart_stopbit* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

In case of *is_uart_one5_stopbit* returns 0, you can't use **1.5-bit** for [Stop Bit]. Otherwise, *set_uart_stopbit* returns -2.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*,
get_uart_min_baudrate, *get_uart_max_baudrate*, *set_uart_baudrate*, *is_uart_8_databit_only*,
is_uart_7_and_8_databit_only, *set_uart_parity*, *set_uart_databit*, *is_uart_one5_stopbit*, *set_uart_stopbit*,
set_uart_flow_control, *set_uart_dtrdsr*, *set_uart_tx_delay*.

Remarks

5.2.7 set_uart_flow_control

```
int set_uart_flow_control(unsigned char *mac_address, int uart_index, int flow_control);
```

Description

set_uart_flow_control changes the parameter of [Flow Control] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int flow_control

Each value is defined as follows:

0

NONE

1

RTS/CTS

2

XON/XOFF

Return values



Each value returned by *set_uart_flow_control* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

In case of CSE-M53 or CSE-H53, if you want to use **XON/XOFF** then [**Baudrate**] should be less than 230,400bps. Otherwise, *set_uart_flow_control* returns -2.

-3

In case of *is_uart_xonxoff* returns 0, you can't use **XON/XOFF** for [**Flow Control**]. Otherwise, *set_uart_flow_control* returns -3.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*,
get_uart_min_baudrate, *get_uart_max_baudrate*, *set_uart_baudrate*, *is_uart_8_databit_only*,
is_uart_7_and_8_databit_only, *set_uart_parity*, *set_uart_databit*, *is_uart_one5_stopbit*, *set_uart_stopbit*,
is_uart_xonxoff, *set_uart_flow_control*, *set_uart_dtrdsr*, *set_uart_tx_delay*.

Remarks

5.2.8 set_uart_dtrdsr

```
int set_uart_dtrdsr(unsigned char *mac_address, int uart_index, int onoff);
```

Description

set_uart_dtrdsr changes the parameter of [**DTR/DSR**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_uart_dtrdsr* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_dtrdsr* returns 0.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*, *get_uart_min_baudrate*, *get_uart_max_baudrate*, *set_uart_baudrate*, *is_uart_8_databit_only*, *is_uart_7_and_8_databit_only*, *set_uart_parity*, *set_uart_databit*, *is_uart_one5_stopbit*, *set_uart_stopbit*, *set_uart_flow_control*, *is_uart_dtrdsr*, *set_uart_dtrdsr*, *set_uart_tx_delay*.

Remarks

set_uart_dtrdsr is only available if *is_uart_dtrdsr* returns 1.

5.2.9 set_uart_tx_delay

```
int set_uart_tx_delay(unsigned char *mac_address, int uart_index, int tx_delay);
```

Description

set_uart_tx_delay changes the parameter of [TX Interval] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int tx_delay

It is a interval for transmitting each data of a COM port designated by *uart_index*.

Return values

Each value returned by *set_uart_tx_delay* is defined as follows:



1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_tx_delay* returns 0.

-3

It returns -3 if *tx_delay* exceeds proper range. The value should be within 0 ~ 25.

Examples

See also

is_uart_rs232, *is_uart_rs485*, *is_uart_rs422*, *set_uart_serial_type*, *is_uart_ttl*, *set_uart_ttl*,
get_uart_min_baudrate, *get_uart_max_baudrate*, *set_uart_baudrate*, *is_uart_8_databit_only*,
is_uart_7_and_8_databit_only, *set_uart_parity*, *set_uart_databit*, *is_uart_one5_stopbit*, *set_uart_stopbit*,
set_uart_flow_control, *is_uart_dtrdsr*, *set_uart_dtrdsr*, *is_uart_tx_delay*, *set_uart_tx_delay*.

Remarks

set_uart_tx_delay is only available if *is_uart_tx_delay* returns 1.

5.3 TCP/IP

5.3.1 *set_uart_communication_mode*

```
int set_uart_communication_mode(unsigned char *mac_address, int uart_index,
                                int communication_mode);
```

Description

set_uart_communication_mode changes the parameter of [**Communication Mode**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int communication_mode

Each value is defined as follows:

0

- 1 T2S – TCP Server
- 2 ATC – AT Command
- 3 COD – TCP Client
- 4 U2S – UDP
- Serial Modbus/TCP

Return values

Each value returned by *set_uart_communication_mode* is defined as follows:

- 1
If the function succeeds, it returns 1.
- 1
If the ezTCP with a *mac_address* is not found, then it returns -1.
- 2
If you want to use **ATC – AT Command, COD – TCP Client, U2S – UDP or Serial Modbus/TCP** then **[Multiple Connection]** should be disabled. Otherwise, *set_uart_communication_mode* returns -2.
- 3
If you want to use **U2S – UDP** then **[SSL]** should be disabled. Otherwise, *set_uart_communication_mode* returns -3.
- 4
If you want to use **ATC – AT Command, COD – TCP Client, U2S – UDP or Serial Modbus/TCP** then **[SSH]** should be disabled. Otherwise, *set_uart_communication_mode* returns -4.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

- T2S – TCP Server is available when *is_uart_tcp_server* returns 1.
- ATC – AT Command is available when *is_uart_at_command* returns 1.
- COD – TCP Client is available when *is_uart_tcp_client* returns 1.
- U2S – UDP is available when *is_uart_udp* returns 1.



Serial Modbus/TCP is available when *is_uart_serial_modbus* returns 1.

5.3.2 set_uart_peer_address

```
int set_uart_peer_address(unsigned char *mac_address, int uart_index,
                          char *peer_address);
```

Description

set_uart_peer_address changes the parameter of [**Peer Address**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

*char *peer_address*

peer_address points to a character string containing an address of remote host.

If [**IPv6**] is enabled then *peer_address* should have **IPv6 address** of remote host. Otherwise, It should have **IPv4 address** or **host name** of remote host.

Return values

Each value returned by *set_uart_peer_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

If [**IPv6**] is enabled then *peer_address* should have valid **IPv6 address** of remote host. Otherwise, *set_uart_peer_address* returns -2.

Examples

```
char *ipv6_peer_address = "2001:DB8::9";
char *ipv4_peer_address = "192.168.0.5";

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
```



```

{
    int res;
    unsigned char *mac_address = out_eztcp_info.mac_address;

    if(get_ip6(mac_address) == 1)
        res = set_uart_peer_address(mac_address, 0, ipv6_peer_address);
    else
        res = set_uart_peer_address(mac_address, 0, ipv4_peer_address);

    if(res == 1)
        printf("Success\r\n");
    else if(res == -1)
        printf("There is no product information of the requested MAC address\r\n");
    else if(res == -2)
        printf("You entered an incorrect IPV6 address.\r\n");
}

```

See also

is_uart_tcp_server, is_uart_at_command, is_uart_tcp_client, is_uart_udp, is_uart_serial_modbus, set_uart_communication_mode, set_uart_peer_address, set_uart_peer_port, set_uart_local_port, set_uart_cod_tcp_server, set_uart_watermark, set_uart_timeout, set_uart_data_frame_interval, set_uart_separator_length, set_uart_separator_type, set_uart_separator, set_uart_tcp_nodelay, set_uart_rfc2217, set_uart_protocol.

Remarks

The length of [**Peer Address**] is 64 bytes except a NULL byte.

5.3.3 set_uart_peer_port

```
int set_uart_peer_port(unsigned char *mac_address, int uart_index, int peer_port);
```

Description

set_uart_peer_port changes the parameter of [**Peer Port**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int peer_port

It is a TCP port number of TCP server that is running on a remote host.

Return values

Each value returned by *set_uart_peer_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

5.3.4 set_uart_local_port

```
int set_uart_local_port(unsigned char *mac_address, int uart_index, int local_port);
```

Description

set_uart_local_port changes the parameter of [Local Port] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int local_port

It is a TCP port number of TCP server that is running on a ezTCP.

Return values

Each value returned by *set_uart_local_port* is defined as follows:

1

If the function succeeds, it returns 1.



-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

If the value of *local_port* is already used for other environmental variable then *set_uart_local_port* returns -2.

-3

If you want to use 23 for [Local Port] then [Telnet] should be disabled. Otherwise, *set_uart_local_port* returns -3.

-4

In cas of I/O products, if the method of making a Modbus/TCP connection is [Passive Connection] then the value of *local_port* should NOT be use for [Local Port] of Modbus/TCP. Otherwise, *set_uart_local_port* returns -4.

-5

In case of I/O product, if [Web(HTTP)] of I/O Controller is selected then the value of *local_port* should NOT be used for [Web(HTTP) port]. Otherwise, *set_uart_local_port* returns -5.

-6

If *local_port* has 50,005 then *set_uart_local_port* returns -6.

-7

If *local_port* has 50,006 then *set_uart_local_port* returns -7.

-8

If *local_port* has 50,007 then *set_uart_local_port* returns -8.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Rermarks

50,005, 50,006 and 50,007 are using for ezManager library. So, these numbers are prohibited to use for environmental variables.

5.3.5 set_uart_cod_tcp_server

```
int set_uart_cod_tcp_server(unsigned char *mac_address, int uart_index, int onoff);
```

Description

set_uart_cod_tcp_server changes the parameter of [TCP Server] of a COM port designated by

uart_index.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_uart_cod_tcp_server* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

[TCP Server] option is only available when *get_uart_communication_mode* returns 2(COD – TCP Client).

5.3.6 set_uart_watermark

```
int set_uart_watermark(unsigned char *mac_address, int uart_index,, int watermark);
```

Description



set_uart_watermark changes the parameter of [Event Byte] or [Block Size(Byte)] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int watermark

It is a Event Byte or Block Size(Byte) of a COM port designated by *uart_index*.

Return values

Each value returned by *set_uart_watermark* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

The *watermark* is using for [Event Byte] when [Communication Mode] is 0(T2S – TCP Server), 1(ATC – AT Command) or 2(COD – TCP Client).

The *watermark* is using for [Block Size(Byte)] when [Communication Mode] is 3(U2S – UDP).

[Event Byte] or [Block Size(Byte)] of a COM port designated by *uart_index* may be changed by another environmental variables.

5.3.7 set_uart_timeout

```
int set_uart_timeout(unsigned char *mac_address, int uart_index, int timeout);
```

Description

set_uart_timeout changes the parameter of [Timeout] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int timeout

It is a Timeout of a COM port designated by *uart_index*.

Return values

Each value returned by *set_uart_timeout* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

[**Timeout**] option is only available when [**Communication Mode**] is 0(T2S – TCP Server), 1(ATC – AT Command) or 2(COD – TCP Client).

[**Timeout**] of a COM port designated by *uart_index* may be changed by another environmental variables.

5.3.8 set_uart_data_frame_interval

```
int set_uart_data_frame_interval(unsigned char *mac_address, int uart_index,
                                int data_frame_interval);
```

Description

set_uart_data_frame_interval changes the parameter of [**Data Frame Interval(10ms)**] of a COM port designated by *uart_index*.

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int data_frame_interval

It is a Data Frame Interval(10ms) of a COM port designated by *uart_index*.

Return values

Each value returned by *set_uart_data_frame_interval* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_data_frame_interval* return 0.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *is_uart_data_frame_interval*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

[**Data Frame Interval(10ms)**] option is NOT available when [**Communication Mode**] is 4(Serial Modbus/TCP).

5.3.9 set_uart_separator_length

```
int set_uart_separator_length(unsigned char *mac_address, int uart_index,
                             int separator_length);
```

Description

set_uart_separator_length changes the parameter of [**Separator Length**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int separator_length

It is a Separator Length of a COM port designated by *uart_index* and its range is 0 to 4.

Return values

Each value returned by *set_uart_separator_length* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_separator* return 0.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*,
set_uart_communication_mode, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*,
set_uart_cod_tcp_server, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*,
set_uart_separator_length, *is_uart_separator*, *set_uart_separator_type*, *set_uart_separator*,
set_uart_tcp_nodelay, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

5.3.10 set_uart_separator_type

```
int set_uart_separator_type(unsigned char *mac_address, int uart_index,
                           int separator_type);
```

Description

set_uart_separator_type changes the parameter of [Separator Operation] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int separator_type

Each value is defined as follows:

0	Transmit Separators
1	Transmit Separators + 1 Byte
2	Transmit Separators + 2 Bytes

Return values

Each value returned by *set_uart_separator_type* is defined as follows:

1	If the function succeeds, it returns 1.
-1	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
-2	It returns -2 if <i>is_uart_separator</i> return 0.
-3	It returns -3 if <i>separator_type</i> is invalid.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *is_uart_separator*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

5.3.11 set_uart_separator

```
int set_uart_separator(unsigned char *mac_address, int uart_index, int separator_index,
                      char *separator);
```

Description

set_uart_separator changes the parameter of [Separator(HEX)] of a COM port designated by

uart_index.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int separator_index

The ordinal number of COM ports. It starts from 0.

*char *separator*

separator points to a character string containing a separator in hexadecimal format.

Return values

Each value returned by *set_uart_separator* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_separator* return 0.

-3

It returns -3 if *separator_index* is invalid.

Examples

```
char *separator = "3A";

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    int res;
    unsigned char *mac_address = out_eztcp_info.mac_address;

    if(is_uart_separaot(mac_address) == 1)
    {
        res = set_uart_separator(mac_address, 0, 0, separator);

        if(res == 1)
```

```

        printf("Success\r\n");
    else if(res == -1)
        printf("There is no product information of the requested MAC
address\r\n");
    else if(res == -2)
        printf("This product can't use separator function.\r\n");
    else if(res == -3)
        printf("You entered an invalid separator.\r\n");
    }
}

```

See also

is_uart_tcp_server, is_uart_at_command, is_uart_tcp_client, is_uart_udp, is_uart_serial_modbus, set_uart_communication_mode, set_uart_peer_address, set_uart_peer_port, set_uart_local_port, set_uart_cod_tcp_server, set_uart_watermark, set_uart_timeout, set_uart_data_frame_interval, set_uart_separator_length, is_uart_separator, set_uart_separator_type, set_uart_separator, set_uart_tcp_nodelay, set_uart_rfc2217, set_uart_protocol.

Remarks

separator is a character string in hexadecimal format. The length of *separator* is 2.

5.3.12 set_uart_tcp_nodelay

```
int set_uart_tcp_nodelay(unsigned char *mac_address, int uart_index, int onoff);
```

Description

set_uart_tcp_nodelay changes the parameter of [**Disable TCP Transmission Delay**] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_uart_tcp_nodelay* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_uart_tcp_nodelay* return 0.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *is_uart_tcp_nodelay*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

5.3.13 set_uart_rfc2217

```
int set_uart_rfc2217(unsigned char *mac_address, int uart_index, int onoff);
```

Description

set_uart_rfc2217 changes the parameter of [Telnet COM Port Control(RFC2217)] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1
Enable

Return values

Each value returned by *set_uart_rfc2217* is defined as follows:

- 1
If the function succeeds, it returns 1.
- 1
If the ezTCP with a *mac_address* is not found, then it returns -1.
- 2
It returns -2 if *is_uart_rfc2217* return 0.
- 3
It returns -3, if *onoff* is 1 and [Multiple Connection] is selected.
- 4
It returns -4, if *onoff* is 1 and [SSL] or [SSH] is selected.

Examples

See also

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *is_uart_rfc2217*, *set_uart_rfc2217*, *set_uart_protocol*.

Remarks

5.3.14 set_uart_protocol

```
int set_uart_protocol(unsigned char *mac_address, int uart_index, int uart_protocol);
```

Description

set_uart_protocol changes the parameter of [Protocol] of a COM port designated by *uart_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

int uart_protocol

Each value is defined as follows:

<i>0</i>	TCP
<i>1</i>	TCP + TELNET
<i>2</i>	TCP + SSL

Return values

Each value returned by *set_uart_protocol* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if <i>is_uart_protocol</i> return 0.

Examples**See also**

is_uart_tcp_server, *is_uart_at_command*, *is_uart_tcp_client*, *is_uart_udp*, *is_uart_serial_modbus*, *set_uart_communication_mode*, *set_uart_peer_address*, *set_uart_peer_port*, *set_uart_local_port*, *set_uart_cod_tcp_server*, *set_uart_watermark*, *set_uart_timeout*, *set_uart_data_frame_interval*, *set_uart_separator_length*, *set_uart_separator_type*, *set_uart_separator*, *set_uart_tcp_nodelay*, *set_uart_rfc2217*, *is_uart_protocol*, *set_uart_protocol*.

Remarks

Currently, this function only supports CSE-T16, CSE-T32 and CSE-T48. A supported products may be added in future. Please refer to product's user manual for more detail information.

5.4 CSC-HR2

5.4.1 set_csc_hr2_communication_mode

```
int set_csc_hr2_communication_mode(unsigned char *mac_address,
                                   int communication_mode);
```

Description

set_csc_hr2_communication_mode changes the parameter of [**Communication Mode**] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int communication_mode

Each value is defined as follows:

0

Automatic

1

Change EZU-100 Firmware

Return values

Each value returned by *set_csc_hr2_communication_mode* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

is_csc_hr2, *set_csc_hr2_communication_mode*, *set_csc_hr2_id*, *set_csc_hr2_network_timeout*,
set_csc_hr2_network_threshold, *set_csc_hr2_server_timeout*, *set_csc_hr2_server_threshold*,
set_csc_hr2_checkpoint, *set_csc_hr2_first_server_address*, *set_csc_hr2_first_server_port*,
set_csc_hr2_second_server_address, *set_csc_hr2_second_server_port*.

Remarks

Please refer to product's user manual for more detail information.

5.4.2 set_csc_hr2_id

```
int set_csc_hr2_id(unsigned char *mac_address, char *csc_hr2_id);
```

Description

set_csc_hr2_id changes the parameter of [**ID of CSC-HR2**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *csc_hr2_id*

csc_hr2_id points to a character string containing a ID of CSC-HR2.

Return values

Each value returned by *set_csc_hr2_id* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

<i>is_csc_hr2,</i>	<i>set_csc_hr2_communication_mode,</i>	<i>set_csc_hr2_id,</i>	<i>set_csc_hr2_network_timeout,</i>
<i>set_csc_hr2_network_threshold,</i>	<i>set_csc_hr2_server_timeout,</i>	<i>set_csc_hr2_server_threshold,</i>	
<i>set_csc_hr2_checkport,</i>	<i>set_csc_hr2_first_server_address,</i>	<i>set_csc_hr2_first_server_port,</i>	
<i>set_csc_hr2_second_server_address,</i>	<i>set_csc_hr2_second_server_port.</i>		

Remarks

The length of [ID of CSC-HR2] is 16 bytes except a NULL byte. Please refer to product's user manual for more detail information.

5.4.3 set_csc_hr2_network_timeout

```
int set_csc_hr2_network_timeout(unsigned char *mac_address, int timeout);
```

Description

set_csc_hr2_network_timeout changes the parameter of [Network Timeout] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int timeout

It is a Network Timeout of CSC-HR2.

Return values

Each value returned by *set_csc_hr2_network_timeout* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

is_csc_hr2, *set_csc_hr2_communication_mode*, *set_csc_hr2_id*, *set_csc_hr2_network_timeout*,
set_csc_hr2_network_threshold, *set_csc_hr2_server_timeout*, *set_csc_hr2_server_threshold*,
set_csc_hr2_checkpoint, *set_csc_hr2_first_server_address*, *set_csc_hr2_first_server_port*,
set_csc_hr2_second_server_address, *set_csc_hr2_second_server_port*.

Remarks

Please refer to product's user manual for more detail information.

5.4.4 set_csc_hr2_network_threshold

```
int set_csc_hr2_network_threshold(unsigned char *mac_address, int threshold);
```

Description

set_csc_hr2_network_threshold changes the parameter of [Network Threshold] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int threshold

It is a Network Threshold of CSC-HR2.

Return values

Each value returned by *set_csc_hr2_network_threshold* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.



-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

is_csc_hr2, *set_csc_hr2_communication_mode*, *set_csc_hr2_id*, *set_csc_hr2_network_timeout*,
set_csc_hr2_network_threshold, *set_csc_hr2_server_timeout*, *set_csc_hr2_server_threshold*,
set_csc_hr2_checkport, *set_csc_hr2_first_server_address*, *set_csc_hr2_first_server_port*,
set_csc_hr2_second_server_address, *set_csc_hr2_second_server_port*.

Remarks

Please refer to product's user manual for more detail information.

5.4.5 set_csc_hr2_server_timeout

```
int set_csc_hr2_server_timeout(unsigned char *mac_address, int timeout);
```

Description

set_csc_hr2_server_timeout changes the parameter of [Server Timeout] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int timeout

It is a Server Timeout of CSC-HR2.

Return values

Each value returned by *set_csc_hr2_server_timeout* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also



is_csc_hr2,	set_csc_hr2_communication_mode,	set_csc_hr2_id,	set_csc_hr2_network_timeout,
set_csc_hr2_network_threshold,	set_csc_hr2_server_timeout,	set_csc_hr2_server_threshold,	
set_csc_hr2_checkport,	set_csc_hr2_first_server_address,	set_csc_hr2_first_server_port,	
set_csc_hr2_second_server_address,	set_csc_hr2_second_server_port.		

Remarks

Please refer to product's user manual for more detail information.

5.4.6 set_csc_hr2_server_threshold

```
int set_csc_hr2_server_threshold(unsigned char *mac_address, int threshold);
```

Description

set_csc_hr2_server_threshold changes the parameter of [Server Threshold] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int threshold

It is a Server Threshold of CSC-HR2.

Return values

Each value returned by *set_csc_hr2_server_threshold* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

is_csc_hr2,	set_csc_hr2_communication_mode,	set_csc_hr2_id,	set_csc_hr2_network_timeout,
set_csc_hr2_network_threshold,	set_csc_hr2_server_timeout,	set_csc_hr2_server_threshold,	
set_csc_hr2_checkport,	set_csc_hr2_first_server_address,	set_csc_hr2_first_server_port,	
set_csc_hr2_second_server_address,	set_csc_hr2_second_server_port.		

Remarks

Please refer to product's user manual for more detail information.

5.4.7 set_csc_hr2_checkport

```
int set_csc_hr2_checkport(unsigned char *mac_address, int check_port);
```

Description

set_csc_hr2_check_port changes the parameter of [**Check Port**] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int check_port

It is a TCP port number of Check Port.

Return values

Each value returned by *set_csc_hr2_checkport* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

<i>is_csc_hr2,</i>	<i>set_csc_hr2_communication_mode,</i>	<i>set_csc_hr2_id,</i>	<i>set_csc_hr2_network_timeout,</i>
<i>set_csc_hr2_network_threshold,</i>	<i>set_csc_hr2_server_timeout,</i>	<i>set_csc_hr2_server_threshold,</i>	
<i>set_csc_hr2_checkport,</i>	<i>set_csc_hr2_first_server_address,</i>	<i>set_csc_hr2_first_server_port,</i>	
<i>set_csc_hr2_second_server_address,</i>	<i>set_csc_hr2_second_server_port.</i>		

Remarks

Please refer to product's user manual for more detail information.

5.4.8 set_csc_hr2_first_server_address

```
int set_csc_hr2_first_server_address(unsigned char *mac_address, char *server_address);
```

Description

set_csc_hr2_first_server_address changes the parameter of [**First Server Address**] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *server_address*

server_address points to a character string containing a First Server Address.

Return values

Each value returned by *set_csc_hr2_first_server_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

<i>is_csc_hr2,</i>	<i>set_csc_hr2_communication_mode,</i>	<i>set_csc_hr2_id,</i>	<i>set_csc_hr2_network_timeout,</i>
<i>set_csc_hr2_network_threshold,</i>	<i>set_csc_hr2_server_timeout,</i>	<i>set_csc_hr2_server_threshold,</i>	
<i>set_csc_hr2_checkport,</i>	<i>set_csc_hr2_first_server_address,</i>	<i>set_csc_hr2_first_server_port,</i>	
<i>set_csc_hr2_second_server_address,</i>	<i>set_csc_hr2_second_server_port.</i>		

Remarks

The length of [**First Server Address**] is 64 bytes except a NULL byte. Please refer to product's user manual for more detail information.

5.4.9 set_csc_hr2_first_server_port

```
int set_csc_hr2_first_server_port(unsigned char *mac_address, int server_port);
```

Description

get_csc_hr2_first_server_port changes the parameter of [**Port Number**] of First Server Address that CSC-HR2 will be connected to.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int server_port

It is a TCP port number of First Server Address.

Return values

Each value returned by *set_csc_hr2_first_server_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

<i>is_csc_hr2</i> ,	<i>set_csc_hr2_communication_mode</i> ,	<i>set_csc_hr2_id</i> ,	<i>set_csc_hr2_network_timeout</i> ,
<i>set_csc_hr2_network_threshold</i> ,	<i>set_csc_hr2_server_timeout</i> ,	<i>set_csc_hr2_server_threshold</i> ,	
<i>set_csc_hr2_checkport</i> ,	<i>set_csc_hr2_first_server_address</i> ,	<i>set_csc_hr2_first_server_port</i> ,	
<i>set_csc_hr2_second_server_address</i> ,	<i>set_csc_hr2_second_server_port</i> .		

Remarks

Please refer to product's user manual for more detail information.

5.4.10 *set_csc_hr2_second_server_address*

```
int set_csc_hr2_second_server_address(unsigned char *mac_address,
                                     char *server_address);
```

Description

set_csc_hr2_second_server_address changes the parameter of [Second Server Address] of CSC-HR2.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *server_address*

server_address points to a character string containing a Second Server Address.

Return values

Each value returned by *set_csc_hr2_second_server_address* is defined as follows:



1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

is_csc_hr2, *set_csc_hr2_communication_mode*, *set_csc_hr2_id*, *set_csc_hr2_network_timeout*,
set_csc_hr2_network_threshold, *set_csc_hr2_server_timeout*, *set_csc_hr2_server_threshold*,
set_csc_hr2_checkpoint, *set_csc_hr2_first_server_address*, *set_csc_hr2_first_server_port*,
set_csc_hr2_second_server_address, *set_csc_hr2_second_server_port*.

Remarks

The length of [**Second Server Address**] is 64 bytes except a NULL byte. Please refer to product's user manual for more detail information.

5.4.11 *set_csc_hr2_second_server_port*

```
int set_csc_hr2_second_server_port(unsigned char *mac_address, int server_port);
```

Description

set_csc_hr2_second_server_port changes the parameter of [**Port Number**] of Second Server Address that CSC-HR2 will be connected to.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int server_port

It is a TCP port number of Second Server Address.

Return values

Each value returned by *set_csc_hr2_second_server_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.



-2

It returns -2 if *is_csc_hr2* return 0.

Examples

See also

<i>is_csc_hr2</i> ,	<i>set_csc_hr2_communication_mode</i> ,	<i>set_csc_hr2_id</i> ,	<i>set_csc_hr2_network_timeout</i> ,
<i>set_csc_hr2_network_threshold</i> ,	<i>set_csc_hr2_server_timeout</i> ,	<i>set_csc_hr2_server_threshold</i> ,	
<i>set_csc_hr2_checkport</i> ,	<i>set_csc_hr2_first_server_address</i> ,	<i>set_csc_hr2_first_server_port</i> ,	
<i>set_csc_hr2_second_server_address</i> ,	<i>set_csc_hr2_second_server_port</i> .		

Remarks

Please refer to product's user manual for more detail information.

5.5 I/O Controller

5.5.1 *set_io_http*

```
int set_io_http(unsigned char *mac_address, int onoff);
```

Description

set_io_http changes the parameter of [Web(HTTP)] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_io_http* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.



-2

It returns -2 if *is_io* return 0.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.2 *set_io_http_port*

```
int set_io_http_port(unsigned char *mac_address, int http_port);
```

Description

set_io_http_port changes the parameter of [**Web(HTTP) port**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int http_port

It is a HTTP port number of [**Web(HTTP) port**].

Return values

Each value returned by *set_io_http_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* return 0.

Examples



See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks**5.5.3 set_io_html_size**

```
int set_io_html_size(unsigned char *mac_address, int html_size);
```

Description

set_io_html_size changes the parameter of [Size of Web(HTTP)] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int html_size

Each value is defined as follows:

<i>0</i>	80KB
<i>1</i>	96KB
<i>2</i>	112KB

Return values

Each value returned by *set_io_html_size* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if <i>is_io</i> return 0.
<i>-3</i>	

It returns -3 if *html_size* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*,
set_io_output_automatic_initialize, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*,
set_io_output_address, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*,
set_io_tcp_type, *set_io_multi_connection_number*, *set_io_modbus_peer_address*,
set_io_modbus_peer_port, *set_io_modbus_local_port*, *set_io_event_notification*,
set_event_notification_email, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*,
set_io_port_macro, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.4 set_io_modbus

```
int set_io_modbus(unsigned char *mac_address, int onoff);
```

Description

set_io_modbus changes the parameter of [Modbus/TCP] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	Enable

Return values

Each value returned by *set_io_modbus* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if <i>is_io</i> return 0.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.5 set_io_input_notification

```
int set_io_input_notification(unsigned char *mac_address, int onoff);
```

Description

set_io_input_notification changes the parameter of [**Notify Input Port Change**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	Enable

Return values

Each value returned by *set_io_input_notification* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if <i>is_io</i> return 0.

Examples



See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.6 set_io_output_automatic_initialize

```
int set_io_output_automatic_initialize(unsigned char *mac_address, int onoff);
```

Description

set_io_output_automatic_initialize changes the parameter of **[Initialize the output port state(The output port will be changed to the[Initial State] when Modbus/TCP is disconnected.)]** of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	Enable

Return values

Each value returned by *set_io_output_automatic_initialize* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if <i>is_io</i> returns 0.
<i>-3</i>	It returns -2 if <i>is_io_output_automatic_initialize</i> returns 0.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, is_io_output_automatic_initialize, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

Please refer to product's user manual for more detail information.

5.5.7 set_io_modbus_type

```
int set_io_modbus_type(unsigned char *mac_address, int modbus_type);
```

Description

set_io_modbus_type changes the parameter of [Master/Slave] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int modbus_type

Each value is defined as follows:

0

Slave

1

Master

Return values

Each value returned by *set_io_modbus_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *modbus_type* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.8 set_io_unit_id

```
int set_io_unit_id(unsigned char *mac_address, int unit_id);
```

Description

set_io_unit_id changes the parameter of [Unit ID] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int unit_id

It is a Unit ID of I/O Controller.

Return values

Each value returned by *set_io_unit_id* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

Examples



See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks**5.5.9 set_io_input_address**

```
int set_io_input_address(unsigned char *mac_address, int input_address);
```

Description

set_io_input_address changes the parameter of [**Input Port Base Address**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int input_address

It is a Input Port Base Address of I/O Controller.

Return values

Each value returned by *set_io_input_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *input_address* is invalid.

Examples**See also**

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address,

set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type,
 set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address,
 set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification,
 set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro,
 set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.10 set_io_output_address

```
int set_io_output_address(unsigned char *mac_address, int output_address);
```

Description

set_io_output_address changes the parameter of [**Output Port Base Address**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int input_address

It is a Output Port Base Address of I/O Controller.

Return values

Each value returned by *set_io_output_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *output_address* is invalid.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification,
 set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address,
 set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type,
 set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address,
 set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification,
 set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro,

set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.11 set_io_poll_interval

```
int set_io_poll_interval(unsigned char *mac_address, int poll_interval);
```

Description

set_io_poll_interval changes the parameter of [**Poll Interval**] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int poll_interval

It is a Poll Interval of I/O Controller.

Return values

Each value returned by *set_io_poll_interval* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.12 set_io_slave_control_type

```
int set_io_slave_control_type(unsigned char *mac_address, int slave_control_type);
```

Description

set_io_slave_control_type changes the parameter of [Control Method of Slave's Output Ports] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int slave_control_type

Each value is defined as follows:

- 0
FC 16 (Multiple)
- 1
FC 05 (Single)

Return values

Each value returned by *set_io_slave_control_type* is defined as follows:

- 1
If the function succeeds, it returns 1.
- 1
If the ezTCP with a *mac_address* is not found, then it returns -1.
- 2
It returns -2 if *is_io* returns 0.
- 3
It returns -3 if *slave_control_type* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks



Please refer to product's user manual for more detail information.

5.5.13 set_io_master_control_type

```
int set_io_master_control_type(unsigned char *mac_address, int master_control_type);
```

Description

set_io_master_control_type changes the parameter of [Control Method of Master's Output Ports] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int master_control_type

Each value is defined as follows:

0

AND

1

OR

Return values

Each value returned by *set_io_master_control_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *master_control_type* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*,
set_io_output_automatic_initialize, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*,
set_io_output_address, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*,
set_io_tcp_type, *set_io_multi_connection_number*, *set_io_modbus_peer_address*,
set_io_modbus_peer_port, *set_io_modbus_local_port*, *set_io_event_notification*,
set_event_notification_email, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*,

set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

Please refer to product's user manual for more detail information.

5.5.14 set_io_tcp_type

```
int set_io_tcp_type(unsigned char *mac_address, int tcp_type);
```

Description

set_io_tcp_type changes method of making a Modbus/TCP connection.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int master_control_type

Each value is defined as follows:

0

Passive Connection

1

Active Connection

Return values

Each value returned by *set_io_tcp_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *tcp_type* is invalid.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type,



set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address,
 set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification,
 set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro,
 set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

Please refer to product's user manual for more detail information.

5.5.15 set_io_multi_connection_number

```
int set_io_multi_connection_number(unsigned char *mac_address, int number);
```

Description

set_io_multi_connection_number changes the maximum number of Modbus/TCP clients that I/O Controllers can accept at the same time if I/O controller is worked as a Passive.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int number

It is a number of TCP clients that I/O Controllers can accept at the same time. Its range is 1 to 8.

Return values

Each value returned by *set_io_multiple_connection_number* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *number* is invalid.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification,
 set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address,
 set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type,
 set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address,

set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification,
 set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro,
 set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.16 set_io_modbus_peer_address

```
int set_io_modbus_peer_address(unsigned char *mac_address, char *peer_address);
```

Description

set_io_modbus_peer_address changes the parameter of [**Peer Address**] of I/O Controllers if it works as a Modbus/TCP client.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *peer_address*

peer_address points to a character string containing an address of remote host.

It should have **IPv4 address** or **host name** of remote host.

Return values

Each value returned by *set_io_modbus_peer_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

Examples

See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification,
 set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address,
 set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type,
 set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address,
 set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification,
 set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro,
 set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

The length of [**Peer Address**] is 64 bytes except a NULL byte.

5.5.17 set_io_modbus_peer_port

```
int set_io_modbus_peer_port(unsigned char *mac_address, int peer_port);
```

Description

set_io_modbus_peer_port changes the parameter of [**Peer Port**] of I/O Controllers if it works as a Modbus/TCP client.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int peer_port

It is a TCP port number of remote host.

Return values

Each value returned by *set_io_modbus_peer_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks



5.5.18 set_io_modbus_local_port

```
int set_io_modbus_local_port(unsigned char *mac_address, int local_port);
```

Description

set_io_modbus_local_port changes the parameter of [Local Port] of I/O Controllers if it works as a Modbus/TCP server.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int peer_port

It is a TCP port number of I/O Controller.

Return values

Each value returned by *set_io_modbus_local_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.19 set_io_event_notification

```
int set_io_event_notification(unsigned char *mac_address, int onoff)
```

Description

set_io_event_notification changes the parameter of [Notify Input or Output Port Change(Email)] of I/O Controller.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_io_event_notification* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *is_event_notification* returns 0.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *is_event_notification*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.20 set_event_notification_email

```
int set_event_notification_email(unsigned char *mac_address, char *email);
```

Description

set_event_notification_email changes the parameter of [Email Address] that is used to send an email if a status of input or output ports has changed.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *email*

email points to a character string containing an email address.

Return values

Each value returned by *set_event_notification_email* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_event_notification* returns 0.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *is_event_notification*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

The length of [Email Address] is 64 bytes include a NULL byte.

5.5.21 set_io_event_notification_port

```
int set_io_event_notification_port(unsigned char *mac_address, int port_flag,
                                  int port_index, int onoff);
```

Description

set_io_event_notification_port changes the parameter of [Notify Input or Output Port Change(Email)] of an input or output port designated by *port_flag* and *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_flag

If *port_flag* is 0 then it means input ports. Otherwise, it means output ports.

int port_index

The ordinal number of input or output ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_io_event_notification_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *is_event_notification* returns 0.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*,
set_io_output_automatic_initialize, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*,

set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type,
 set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address,
 set_io_modbus_peer_port, set_io_modbus_local_port, is_event_notification, set_io_event_notification,
 set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro,
 set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.22 set_io_input_valid_time

```
int set_io_input_valid_time(unsigned char *mac_address, int port_index,  
                           int input_valid_time);
```

Description

set_io_input_valid_time changes the parameter of [Valid Time(ms)] of an input port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of input ports. It starts from 0.

int input_valid_time

It is a Valid Time(ms) of input port designated by *port_index*.

Return values

Each value returned by *set_io_input_valid_time* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *input_valid_time* is invalid.

Examples

See also



is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.23 set_io_macro

```
int set_io_macro(unsigned char *mac_address, int onoff);
```

Description

set_io_macro changes the parameter of [Macro] of I/O Controller. This function is valid if *get_io_macro_type* returns 0.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_io_macro* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *get_io_macro_type* returns 1.

Examples



See also

is_io, set_io_http, set_io_http_port, set_io_html_size, set_io_modbus, set_io_input_notification, set_io_output_automatic_initialize, set_io_modbus_type, set_io_unit_id, set_io_input_address, set_io_output_address, set_io_poll_interval, set_io_slave_control_type, set_io_master_control_type, set_io_tcp_type, set_io_multi_connection_number, set_io_modbus_peer_address, set_io_modbus_peer_port, set_io_modbus_local_port, set_io_event_notification, set_event_notification_email, set_io_event_notification_port, set_io_input_valid_time, set_io_macro, set_io_port_macro, set_io_macro_text, set_io_output_delay, set_io_output_initial_state, set_io_comment.

Remarks

5.5.24 set_io_port_macro

```
int set_io_port_macro(unsigned char *mac_address, int port_index, int onoff);
```

Description

set_io_port_macro changes the parameter of [Macro] of an output port designated by *port_index*. This function is valid if *get_io_macro_type* returns 1.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_io_macro_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *get_io_macro_type* returns 0.

-4

It returns -3 if *port_index* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.25 set_io_macro_text

```
int set_io_macro_text(unsigned char *mac_address, int port_index, char *macro_text);
```

Description

set_io_macro_text changes the parameter of Macro text of an output port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

*char *macro_text*

macro_text points to a character string containing a Macro for an output port designated by *port_index*.

Return values

Each value returned by *set_io_macro_text* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2



It returns -2 if *is_io* returns 0.

-3

It returns -3 if *macro_text* has a syntactic error.

-4

It returns -4 if *port_index* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

The length of Macro text is 32 bytes except a NULL byte. **IO_SCRIPT_LEN** defines the maximum length of Macro text.

5.5.26 *set_io_output_delay*

```
int set_io_output_delay(unsigned char *mac_address, int port_index, int output_delay);
```

Description

set_io_output_delay changes the parameter of [**Delay(ms)**] of an output port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

int output_delay

It is a Delay(ms) of output port designated by *port_index*.

Return values

Each value returned by *set_io_output_delay* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *port_index* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*,
set_io_output_automatic_initialize, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*,
set_io_output_address, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*,
set_io_tcp_type, *set_io_multi_connection_number*, *set_io_modbus_peer_address*,
set_io_modbus_peer_port, *set_io_modbus_local_port*, *set_io_event_notification*,
set_event_notification_email, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*,
set_io_port_macro, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.27 *set_io_output_initial_state*

```
int set_io_output_initial_state(unsigned char *mac_address, int port_index, int onoff);
```

Description

set_io_output_initial_state changes the parameter of [Initial State] of an output port designated by *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_index

The ordinal number of output ports. It starts from 0.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_io_output_initial_state* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *port_index* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*,
set_io_output_automatic_initialize, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*,
set_io_output_address, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*,
set_io_tcp_type, *set_io_multi_connection_number*, *set_io_modbus_peer_address*,
set_io_modbus_peer_port, *set_io_modbus_local_port*, *set_io_event_notification*,
set_event_notification_email, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*,
set_io_port_macro, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

5.5.28 set_io_comment

```
int set_io_comment(unsigned char *mac_address, int port_flag, int port_index,
                  char *comment);
```

Description

set_io_comment changes the parameter of comment of an input or output port designated by *port_flag* and *port_index*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int port_flag

If *port_flag* is 0 then it means input ports. Otherwise, it means output ports.

int port_index

The ordinal number of input or output ports. It starts from 0.

*char *comment*

comment points to a character string containing a comment for an input or output port designated by *port_index*.

Return values

Each value returned by *set_io_comment* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_io* returns 0.

-3

It returns -3 if *port_index* is invalid.

Examples

See also

is_io, *set_io_http*, *set_io_http_port*, *set_io_html_size*, *set_io_modbus*, *set_io_input_notification*, *set_io_output_automatic_initialize*, *set_io_modbus_type*, *set_io_unit_id*, *set_io_input_address*, *set_io_output_address*, *set_io_poll_interval*, *set_io_slave_control_type*, *set_io_master_control_type*, *set_io_tcp_type*, *set_io_multi_connection_number*, *set_io_modbus_peer_address*, *set_io_modbus_peer_port*, *set_io_modbus_local_port*, *set_io_event_notification*, *set_event_notification_email*, *set_io_event_notification_port*, *set_io_input_valid_time*, *set_io_macro*, *set_io_port_macro*, *set_io_macro_text*, *set_io_output_delay*, *set_io_output_initial_state*, *set_io_comment*.

Remarks

The length of comment is 16 bytes except a NULL byte. **IO_COMMENT_LEN** defines the maximum length of comment.

5.6 Wireless Network

5.6.1 set_wlan_type

```
int set_wlan_type(unsigned char *mac_address, int wlan_type);
```

Description

set_wlan_type changes the parameter of [WLAN Topology].

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int wlan_type

Each value is defined as follows:

<i>0</i>	Ad-Hoc
<i>1</i>	Infrastructure
<i>2</i>	Soft AP

Return values

Each value returned by *set_wlan_type* is defined as follows:

- 1*
If the function succeeds, it returns 1.
- 1*
If the ezTCP with a *mac_address* is not found, then it returns -1.
- 2*
It returns -2 if *is_wlan* returns 0.
- 3*
It returns -3 if *wlan_type* is invalid.
- 4*
If you want to use **Soft AP(2)** for [WLAN Topology] then *is_wlan_soft_ap* should return 1. Otherwise, *set_wlan_type* returns -4.

Examples

See also

is_wlan, *is_wlan_soft_ap*, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

5.6.2 set_wlan_channel

```
int set_wlan_channel(unsigned char *mac_address, int channel);
```

Description

set_wlan_channel changes the parameter of [Channel].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int channel

It is a number of Channel.

Return values

Each value returned by *set_wlan_channel* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*,
set_wlan_encryption_type, *set_wlan_authentication_type*, *set_wlan_wep_key_length*,
set_wlan_wep_key_index, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*,
set_wlan_shared_key, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*,
set_wlan_wpa_enterprise_pwd.

Remarks

The [Channel] is using if [WLAN Topology] is **Ad-Hoc or Soft-AP**.

5.6.3 set_wlan_ssid

```
int set_wlan_ssid(unsigned char *mac_address, char *ssid);
```

Description

set_wlan_ssid changes the parameter of [SSID].

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ssid*

ssid points to a character string containing a SSID of an access point.

Return values

Each value returned by *set_wlan_ssid* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*,
set_wlan_encryption_type, *set_wlan_authentication_type*, *set_wlan_wep_key_length*,
set_wlan_wep_key_index, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*,
set_wlan_shared_key, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*,
set_wlan_wpa_enterprise_pwd.

Remarks

The length of [SSID] is 32 bytes except a NULL byte.

5.6.4 set_wlan_antenna

```
int set_wlan_antenna(unsigned char *mac_address, int antenna_type);
```

Description

set_wlan_antenna changes the parameter of [Antenna] option.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int antenna_type

Each value is defined as follows:

0	Internal Antenna
1	External Antenna

Return values

Each value returned by *set_wlan_antenna* is defined as follows:

1	If the function succeeds, it returns 1.
-1	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
-2	It returns -2 if <i>is_wlan</i> returns 0.
-3	It returns -3 if <i>is_wlan_antenna</i> returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *is_wlan_antenna*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

Please refer to product's user manual for more detail information.

5.6.5 set_wlan_phy_mode

```
int set_wlan_phy_mode(unsigned char *mac_address, int phy_mode);
```

Description

set_wlan_phy_mode changes the parameter of [**Phy Mode**] in the wireless LAN [**Advanced Settings**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int phy_mode

Each value is defined as follows:

1	802.11
2	802.11b
3	802.11b/g

Return values

Each value returned by *set_wlan_phy_mode* is defined as follows:

1	If the function succeeds, it returns 1.
-1	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
-2	It returns -2 if <i>is_wlan</i> returns 0.
-3	It returns -3 if <i>is_wlan_phy_mode</i> returns 0.
-4	It returns -4 if <i>phy_mode</i> is invalid.

Examples**See also**

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *is_wlan_phy_mode*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

Please refer to product's user manual for more detail information.

5.6.6 set_wlan_short_preamble

```
int set_wlan_short_preamble(unsigned char *mac_address, int onoff);
```

Description

set_wlan_short_preamble changes the parameter of [**Short Preamble**] in the wireless LAN [**Advanced Settings**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_wlan_short_preamble* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_phy_mode* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *is_wlan_phy_mode*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

[**Short Preamble**] is using if [**Phy Mode**] is **2(802.11b)** or **3(802.11b/g)**. Please refer to product's user manual for more detail information.

5.6.7 set_wlan_short_slot

```
int set_wlan_short_slot(unsigned char *mac_address, int onoff);
```

Description

set_wlan_short_slot changes the parameter of [Short Slot] in the wireless LAN [Advanced Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_wlan_short_slot* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_phy_mode* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *is_wlan_phy_mode*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

[Short Preamble] is using if [Phy Mode] is 3(802.11b/g). Please refer to product's user manual for more

detail information.

5.6.8 set_wlan_cts_protection

```
int set_wlan_cts_protection(unsigned char *mac_address, int onoff);
```

Description

set_wlan_cts_protection changes the parameter of [CTS Protection] in the wireless LAN [Advanced Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_wlan_cts_protection* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_phy_mode* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *is_wlan_phy_mode*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

[**Short Preamble**] is using if [**Phy Mode**] is **3(802.11b/g)**. Please refer to product's user manual for more detail information.

5.6.9 set_wlan_background_scan

```
int set_wlan_background_scan(unsigned char *mac_address, int onoff);
```

Description

set_wlan_background_scan changes the parameter of [**Background Scan**] in the wireless LAN [**Advanced Settings**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_wlan_background_scan* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_background_scan* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *is_wlan_background_scan*,
set_wlan_background_scan, *set_wlan_encryption_type*, *set_wlan_authentication_type*,

set_wlan_wep_key_length, set_wlan_wep_key_index, set_wlan_wep_key_data_type, set_wlan_wep_key, set_wlan_wpa_passphrase, set_wlan_shared_key, set_wlan_wpa_enterprise, set_wlan_wpa_enterprise_id, set_wlan_wpa_enterprise_pwd.

Remarks

Please refer to product's user manual for more detail information.

5.6.10 set_wlan_encryption_type

```
int set_wlan_encryption_type(unsigned char *mac_address, int encryption_type);
```

Description

set_wlan_encryption_type changes the parameter of [Encryption] in the wireless LAN [Security Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int encryption_type

Each value is defined as follows:

0

없음

1

WEP

2

WPA

Return values

Each value returned by *set_wlan_encryption_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_rsn* returns 1.

-4

It returns -4 if *encryption_type* is invalid.

-5

It returns -5 if *encryption_type* is **WPA(2)** and [WLAN Topology] is **Ad-Hoc(0)**.

Examples

See also

is_wlan, set_wlan_type, set_wlan_channel, set_wlan_ssid, set_wlan_antenna, set_wlan_phy_mode, set_wlan_short_preamble, set_wlan_short_slot, set_wlan_cts_protection, set_wlan_background_scan, is_wlan_rsn, set_wlan_encryption_type, set_wlan_authentication_type, set_wlan_wep_key_length, set_wlan_wep_key_index, set_wlan_wep_key_data_type, set_wlan_wep_key, set_wlan_wpa_passphrase, set_wlan_shared_key, set_wlan_wpa_enterprise, set_wlan_wpa_enterprise_id, set_wlan_wpa_enterprise_pwd.

Remarks

5.6.11 set_wlan_authentication_type

```
int set_wlan_authentication_type(unsigned char *mac_address, int authentication_type);
```

Description

set_wlan_authentication_type changes the parameter of [Authentication] or WPA [Authentication / Encryption] in the wireless LAN [Security Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int encryption_type

If *get_wlan_encryption_type* returns **0(NONE)** then *authentication_type* is defined as follows:

1

Open System

If *get_wlan_encryption_type* returns **1(WEP)** then *authentication_type* is defined as follows:

1

Open System

2

Shared Key

3

Auto

If *get_wlan_encryption_type* returns **2(WPA)** then *authentication_type* is defined as follows:



0	WPA PSK – TKIP
1	WPA PSK – AES
2	WPA PSK – TKIP/AES
3	WPA2 PSK – TKIP
4	WPA2 PSK – AES
5	WPA2 PSK- TKIP/AES

Return values

Each value returned by *set_wlan_authentication_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_rsn* returns 1.

-4

It returns -4 if *authentication_type* is invalid when **[Encryption]** in the wireless LAN **[Security Settings]** is **0(NONE)** or **1(WEP)**.

-5

It returns -5 if *authentication_type* is invalid when **[Encryption]** in the wireless LAN **[Security Settings]** is **2(WPA)**.

If *is_wlan_authentication_type* returns 0 then *authentication_type* should be one of **0(WPA PSK - TKIP)** or **4(WPA2 PSK – AES)**.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *is_wlan_rsn*, *set_wlan_encryption_type*, *is_wlan_authentication_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

5.6.12 set_wlan_wep_key_length

```
int set_wlan_wep_key_length(unsigned char *mac_address, int wep_key_length);
```

Description

set_wlan_wep_key_length changes length of WEP key.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int wep_key_length

Each value is defined as follows:

- | | |
|---|---------|
| 1 | 64-bit |
| 2 | 128-bit |

Return values

Each value returned by *set_wlan_wep_key_length* is defined as follows:

- | | |
|----|--|
| 1 | If the function succeeds, it returns 1. |
| -1 | If the ezTCP with a <i>mac_address</i> is not found, then it returns -1. |
| -2 | It returns -2 if <i>is_wlan</i> returns 0. |
| -3 | It returns -3 if <i>wep_key_length</i> is invalid. |

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

5.6.13 set_wlan_wep_key_index

```
int set_wlan_wep_key_index(unsigned char *mac_address, int wep_key_index);
```

Description

set_wlan_wep_key_index changes index of WEP key.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int wep_key_index

The ordinal number of WEP key. Its range is 0 to 3.

Return values

Each value returned by *set_wlan_wep_key_index* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *wep_key_index* is invalid.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*,
set_wlan_encryption_type, *set_wlan_authentication_type*, *set_wlan_wep_key_length*,
set_wlan_wep_key_index, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*,
set_wlan_shared_key, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*,
set_wlan_wpa_enterprise_pwd.

Remarks



5.6.14 set_wlan_wep_key_data_type

```
int set_wlan_wep_key_data_type(unsigned char *mac_address, int wep_key_data_type);
```

Description

set_wlan_wep_key_data_type changes data type of WEP key.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int wep_key_data_type

Each value is defined as follows:

0

HEX

1

ASCII

Return values

Each value returned by *set_wlan_wep_key_data_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *wep_key_data_type* is invalid.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*,
set_wlan_encryption_type, *set_wlan_authentication_type*, *set_wlan_wep_key_length*,
set_wlan_wep_key_index, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*,
set_wlan_shared_key, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*,
set_wlan_wpa_enterprise_pwd.

Remarks



5.6.15 set_wlan_wep_key

```
int set_wlan_wep_key(unsigned char *mac_address, char *wep_key);
```

Description

set_wlan_wep_key changes the value of WEP key.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *wep_key*

wep_key points to a character string containing a WEP key.

Return values

Each value returned by *set_wlan_wep_key* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if the length of *wep_key* is invalid.

-4

It returns -4 if *wep_key* have invalid characters.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    int wep_key_length = 1; // 64bit
    int wep_key_index = 1; // index #1
    int wep_key_data_type = 1; // ASCII
    char *wep_key = "12345";
```

```

    if(is_wlan(mac_address) == 1)
    {
        set_wlan_wep_key_length(mac_address, wep_key_length);
        set_wlan_wep_key_index(mac_address, wep_key_index);
        set_wlan_wep_key_data_type(mac_address, wep_key_data_type);
        set_wlan_wep_key(mac_address, wep_key);
    }
}

```

See also

is_wlan, set_wlan_type, set_wlan_channel, set_wlan_ssid, set_wlan_antenna, set_wlan_phy_mode, set_wlan_short_preamble, set_wlan_short_slot, set_wlan_cts_protection, set_wlan_background_scan, set_wlan_encryption_type, set_wlan_authentication_type, set_wlan_wep_key_length, set_wlan_wep_key_index, set_wlan_wep_key_data_type, set_wlan_wep_key, set_wlan_wpa_passphrase, set_wlan_shared_key, set_wlan_wpa_enterprise, set_wlan_wpa_enterprise_id, set_wlan_wpa_enterprise_pwd.

Remarks

If WEP key length is **64-bit** then *wep_key* should have **10 characters in HEX(0 ~ 9, A ~ F or a ~ f)** or **5 characters in ASCII**.

If WEP key length is **128-bit** then *wep_key* should have **26 characters in HEX(0 ~ 9, A ~ F or a ~ f)** or **13 characters in ASCII**.

5.6.16 set_wlan_wpa_passphrase

```
int set_wlan_wpa_passphrase(unsigned char *mac_address, char *wpa_pp);
```

Description

set_wlan_wpa_passphrase changes WPA passphrase.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *wpa_pp*

wpa_pp points to a character string containing a WPA passphrase.

Return values

Each value returned by *set_wlan_wpa_passphrase* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_rsn* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*,
set_wlan_encryption_type, *set_wlan_authentication_type*, *set_wlan_wep_key_length*,
set_wlan_wep_key_index, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*,
set_wlan_shared_key, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*,
set_wlan_wpa_enterprise_pwd.

Remarks

The length of WPA passphrase is 64 bytes include a NULL byte.

5.6.17 set_wlan_shared_key

```
int set_wlan_shared_key(unsigned char *mac_address, char *shared_key);
```

Description

set_wlan_shared_key changes the parameter of [Shared Key].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *shared_key*

shared_key points to a character string containing a Shared Key.

Return values

Each value returned by *set_wlan_shared_key* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2



It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wlan_rsn* returns 0.

-4

It returns -4 if *shared_key* can't be used for WEP key when [WLAN Topology] is **0(Ad-Hoc)** or **2(Soft-AP)**.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*,
set_wlan_short_preamble, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*,
set_wlan_encryption_type, *set_wlan_authentication_type*, *set_wlan_wep_key_length*,
set_wlan_wep_key_index, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*,
set_wlan_shared_key, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*,
set_wlan_wpa_enterprise_pwd.

Remarks

The length of [Shared Key] is 64 bytes include a NULL byte. Please refer to product's user manual for more detail information.

5.6.18 set_wlan_wpa_enterprise

```
int set_wlan_wpa_enterprise(unsigned char *mac_address, int enterprise_type);
```

Description

set_wlan_wpa_enterprise changes the parameter of [802.1X] in the wireless LAN [Security Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int enterprise_type

Each value is defined as follows:

0

Disable

1

EAP TLS

2

EAP TTLS

3

PEAP

Return values

Each value returned by *set_wlan_wpa_enterprise* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wpa_enterprise* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *is_wpa_enterprise*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

Please refer to product's user manual for more detail information.

5.6.19 set_wlan_wpa_enterprise_id

```
int set_wlan_wpa_enterprise_id(unsigned char *mac_address, char *enterprise_id);
```

Description

set_wlan_wpa_enterprise_id changes the parameter of [ID] of 802.1X in the wireless LAN [Security Settings].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *enterprise_id*

enterprise_id points to a character string containing an ID of 802.1X.

Return values

Each value returned by *set_wlan_wpa_enterprise_id* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wpa_enterprise* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *is_wpa_enterprise*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

The length of [**ID**] is 32 bytes except a NULL byte. Please refer to product's user manual for more detail information.

5.6.20 set_wlan_wpa_enterprise_pwd

```
int set_wlan_wpa_enterprise_pwd(unsigned char *mac_address, char *enterprise_pwd);
```

Description

set_wlan_wpa_enterprise_pwd changes the parameter of [**Password**] of 802.1X in the wireless LAN [**Security Settings**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *enterprise_pwd*

enterprise_pwd points to a character string containing an Password of 802.1X.

Return values



Each value returned by *set_wlan_enterprise_pwd* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_wlan* returns 0.

-3

It returns -3 if *is_wpa_enterprise* returns 0.

Examples

See also

is_wlan, *set_wlan_type*, *set_wlan_channel*, *set_wlan_ssid*, *set_wlan_antenna*, *set_wlan_phy_mode*, *set_wlan_short_preamble*, *set_wlan_short_slot*, *set_wlan_cts_protection*, *set_wlan_background_scan*, *set_wlan_encryption_type*, *set_wlan_authentication_type*, *set_wlan_wep_key_length*, *set_wlan_wep_key_index*, *set_wlan_wep_key_data_type*, *set_wlan_wep_key*, *set_wlan_wpa_passphrase*, *set_wlan_shared_key*, *is_wpa_enterprise*, *set_wlan_wpa_enterprise*, *set_wlan_wpa_enterprise_id*, *set_wlan_wpa_enterprise_pwd*.

Remarks

The length of **[Password]** is 32 bytes except a NULL byte.

[Password] of 802.1X in the wireless LAN **[Security Settings]** is NOT available when *get_wlan_wpa_enterprise* returns **0(Disable)** or **1(EAP TLS)**.

Please refer to product's user manual for more detail information.

5.7 Options

5.7.1 set_telnet

```
int set_telnet(unsigned char *mac_address, int onoff);
```

Description

set_telnet changes the parameter of **[Telnet]**.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	Enable

Return values

Each value returned by *set_telnet* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.

Examples

See also

set_telnet, set_send_mac_address, set_ssl, set_ssh, set_ip4_address_search, set_remote_debug, set_tcp_multi_connection, set_power_management, set_comment.

Remarks

5.7.2 set_send_mac_address

```
int set_send_mac_address(unsigned char *mac_address, int onoff);
```

Description

set_send_mac_address changes the parameter of [Send MAC Address].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	Enable

Return values



Each value returned by *set_send_mac_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_send_mac_address* returns 0.

-3

It returns -3 if an ezTCP designated by *mac_address* can't enable [Send MAC Address] and [SSL] at the same time.

-4

It returns -4 if [SSH] is enabled.

Examples

See also

set_telnet, *is_send_mac_address*, *set_send_mac_address*, *set_ssl*, *set_ssh*, *set_ip4_address_search*, *set_remote_debug*, *set_tcp_multi_connection*, *set_power_management*, *set_comment*.

Remarks

5.7.3 set_ssl

```
int set_ssl(unsigned char *mac_address, int onoff);
```

Description

set_ssl changes the parameter of [SSL].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values



Each value returned by *set_ssl* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_ssl* returns 0.

-3

It returns -3 if [SSH] is enabled.

-4

It returns -4 if an ezTCP designated by *mac_address* can't enable [SSL] and [Send MAC Address] at the same time.

-5

It returns -5 if an ezTCP designated by *mac_address* can't use [SSL] with RS-485 or RS-422.

-6

It returns -6 if [Communication Mode] of a COM port is U2S – UDP.

-7

It returns -7 if [IPv6] is enabled.

-8

It returns -8 if [Telnet COM Port Control(RFC2217)] of a COM port is enabled.

Examples

See also

set_telnet, *set_send_mac_address*, *is_ssl*, *set_ssl*, *set_ssh*, *set_ip4_address_search*, *set_remote_debug*, *set_tcp_multi_connection*, *set_power_management*, *set_comment*.

Remarks

5.7.4 set_ssh

```
int set_ssh(unsigned char *mac_address, int onoff);
```

Description

set_ssh changes the parameter of [SSH].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	Enable

Return values

Each value returned by *set_ssh* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if <i>is_ssh</i> returns 0.
<i>-3</i>	It returns -3 if [SSL] is enabled.
<i>-4</i>	It returns -4 if [Serial Type] of a COM port is RS-485 or RS-422.
<i>-5</i>	It returns -5 if [Communication Mode] of a COM port is not T2S – TCP Server.
<i>-6</i>	It returns -6 if [Send MAC Address] is enabled.
<i>-7</i>	It returns -7 if [Telnet COM Port Control(RFC2217)] of a COM port is enabled.

Examples

See also

set_telnet, *set_send_mac_address*, *set_ssl*, *is_ssh*, *set_ssh*, *set_ip4_address_search*, *set_remote_debug*, *set_tcp_multi_connection*, *set_power_management*, *set_comment*.

Remarks

5.7.5 set_ip4_address_search

```
int set_ip4_address_search(unsigned char *mac_address, int onoff);
```

Description

set_ip4_address_search changes the parameter of [IPv4 Address Search].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_ip4_address_search* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

set_telnet, *set_send_mac_address*, *set_ssl*, *set_ssh*, *set_ip4_address_search*, *set_remote_debug*,
set_tcp_multi_connection, *set_power_management*, *set_comment*.

Remarks**5.7.6 set_remote_debug**

```
int set_remote_debug(unsigned char *mac_address, int onoff);
```

Description

set_remote_debug changes the parameter of [Debugging Message].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_remote_debug* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_remote_debug* returns 0.

Examples

See also

set_telnet, *set_send_mac_address*, *set_ssl*, *set_ssh*, *set_ip4_address_search*, *is_remote_debug*, *set_remote_debug*, *set_tcp_multi_connection*, *set_power_management*, *set_comment*.

Remarks

5.7.7 set_tcp_multi_connection

```
int set_tcp_multi_connection(unsigned char *mac_address, int onoff);
```

Description

set_tcp_multi_connection changes the parameter of [Multiple Connection].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable



1
Enable

Return values

Each value returned by *set_tcp_multi_connection* is defined as follows:

- 1
If the function succeeds, it returns 1.
- 1
If the ezTCP with a *mac_address* is not found, then it returns -1.
- 2
It returns -2 if *is_tcp_multi_connection* returns 0.
- 3
It returns -3 if [**Communication Mode**] of a COM port is not T2S – TCP Server.
- 4
It returns -4 if [**Send MAC Address**] is enabled.
- 5
It returns -5 if [**SSL**] is enabled.
- 6
It returns -6 if [**SSH**] is enabled.

Examples

See also

set_telnet, set_send_mac_address, set_ssl, set_ssh, set_ip4_address_search, set_remote_debug, is_tcp_multi_connection, set_tcp_multi_connection, set_power_management, set_comment.

Remarks

5.7.8 set_power_management

```
int set_power_management(unsigned char *mac_address, int onoff);
```

Description

set_power_management changes the parameter of [**Power Management**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_power_management* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if *is_power_management* returns 0.

Examples

See also

set_telnet, *set_send_mac_address*, *set_ssl*, *set_ssh*, *set_ip4_address_search*, *set_remote_debug*, *set_tcp_multi_connection*, *is_power_management*, *set_power_management*, *set_comment*.

Remarks

5.7.9 set_comment

```
int set_comment(unsigned char *mac_address, char *comment);
```

Description

set_comment changes the parameter of [**Comment**].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *comment*

comment points to a character string containing an Comment.

Return values



Each value returned by *set_comment* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

set_telnet, *set_send_mac_address*, *set_ssl*, *set_ssh*, *set_ip4_address_search*, *set_remote_debug*, *set_tcp_multi_connection*, *set_power_management*, *set_comment*.

Remarks

The length of [**Comment**] is 64 bytes except a NULL byte.

5.7.10 set_allowed_mac_address

```
int set_allowed_mac_address(unsigned char *mac_address, int index,
                           char *in_mac_address);
```

Description

set_allowed_mac_address changes the parameter of [**Allowed MAC Address**]. [**Allowed MAC Address**] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int index

The ordinal number of MAC address. The range of *mac_index* is 0 to 5.

Return values

Each value returned by *set_allowed_mac_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if the value of *index* is invalid.

-3

It returns -3 if the value of *in_mac_address* is invalid.

Examples

See also

set_allowed_mac_address, set_allowed_ip4_address, set_allowed_ip4_network_mask_type,
set_allowed_ezmanager, set_allowed_ip6_address, set_allowed_ip6_subnet_prefix_length.

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

5.7.11 set_allowed_ip4_address

```
int set_allowed_ip4_address(unsigned char *mac_address, char *ip4_address);
```

Description

set_allowed_ip4_address changes the parameter of [IPv4 Address] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip4_address*

ip4_address points to a character string containing an IPv4 Address.

Return values

Each value returned by *set_allowed_ip4_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

set_allowed_mac_address, set_allowed_ip4_address, set_allowed_ip4_network_mask_type,
set_allowed_ezmanager, set_allowed_ip6_address, set_allowed_ip6_subnet_prefix_length.

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

5.7.12 set_allowed_ip4_network_mask_type

```
int set_allowed_ip4_network_mask_type(unsigned char *mac_address,
                                     int network_mask_type);
```

Description

set_allowed_ip4_network_mask_type changes the parameter of [Network Mask] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int network_mask_type

Each value is defined as follows:

<i>0</i>	255.255.255.255
<i>1</i>	255.255.255.0
<i>2</i>	255.255.0.0
<i>3</i>	255.0.0.0
<i>4</i>	0.0.0.0

Return values

Each value returned by *set_allowed_ip4_network_mask_type* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if the value of <i>network_mask_type</i> is invalid.

Examples



See also

set_allowed_mac_address, set_allowed_ip4_address, set_allowed_ip4_network_mask_type,
 set_allowed_ezmanager, set_allowed_ip6_address, set_allowed_ip6_subnet_prefix_length.

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

5.7.13 set_allowed_ezmanager

```
int set_allowed_ezmanager(unsigned char *mac_address, int onoff);
```

Description

set_allowed_ezmanager changes the parameter of [Apply To ezManager]. [Apply To ezManager] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int onoff

Each value is defined as follows:

0

Disable

1

Enable

Return values

Each value returned by *set_allowed_ezmanager* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

set_allowed_mac_address, set_allowed_ip4_address, set_allowed_ip4_network_mask_type,
 set_allowed_ezmanager, set_allowed_ip6_address, set_allowed_ip6_subnet_prefix_length.

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

5.7.14 set_allowed_ip6_address

```
int set_allowed_ip6_address(unsigned char *mac_address, char *ip6_address);
```

Description

set_allowed_ip6_address changes the parameter of [IPv6 Address] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ip6_address*

ip6_address points to a character string containing an IPv6 Address.

Return values

Each value returned by *set_allowed_ip6_address* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if the value of *ip6_address* is invalid.

Examples

See also

set_allowed_mac_address, *set_allowed_ip4_address*, *set_allowed_ip4_network_mask_type*,
set_allowed_ezmanager, *set_allowed_ip6_address*, *set_allowed_ip6_subnet_prefix_length*.

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

5.7.15 set_allowed_ip6_subnet_prefix_length

```
int set_allowed_ip6_subnet_prefix_length(unsigned char *mac_address,  
int ip6_subnet_prefix_length);
```

Description

set_allowed_ip6_subnet_prefix_length changes subnet prefix length of [IPv6 Address] in [Allowed IP Range] section. [Allowed IP Range] is one of ezTCP Firewall functions.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int ip6_subnet_prefix_length

ip6_subnet_prefix_length is subnet prefix length of IPv6 address and its range is 0 to 128.

Return values

Each value returned by *set_allowed_ip6_subnet_prefix_length* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if the value of *subnet_prefix_length* is invalid.

Examples

See also

set_allowed_mac_address, *set_allowed_ip4_address*, *set_allowed_ip4_network_mask_type*,
set_allowed_ezmanager, *set_allowed_ip6_address*, *set_allowed_ip6_subnet_prefix_length*.

Remarks

Please refer to product's user manual for more detail information about [ezTCP Firewall].

5.7.16 set_ip4_change_notification_type

```
int set_ip4_change_notification_type(unsigned char *mac_address, int notification_type);
```

Description

set_ip4_change_notification_type changes the parameter of [Protocol] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



int notification_type

Each value is defined as follows:

<i>0</i>	Disable
<i>1</i>	DDNS(dyndns.org)
<i>2</i>	TCP
<i>3</i>	UDP

Return values

Each value returned by *set_ip4_change_notification_type* is defined as follows:

<i>1</i>	If the function succeeds, it returns 1.
<i>-1</i>	If the ezTCP with a <i>mac_address</i> is not found, then it returns -1.
<i>-2</i>	It returns -2 if the value of <i>notification_type</i> is invalid.

Examples**See also**

set_ip4_change_notification_type,
set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_data_type,
set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

5.7.17 set_ip4_change_notification_data_type

```
int set_ip4_change_notification_data_type(unsigned char *mac_address, int data_type);
```

Description

set_ip4_change_notification_data_type changes the parameter of [Data Type] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int notification_type

Each value is defined as follows:

0

ASCII

1

HEX

Return values

Each value returned by *set_ip4_change_notification_data_type* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if the value of *data_type* is invalid.

Examples

See also

set_ip4_change_notification_type,
set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_data_type,
set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

[Data Type] in [Notify IPv4 Change] section is used when [Protocol] is 2(TCP) or 3(UDP).

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

5.7.18 set_ip4_change_notification_interval

```
int set_ip4_change_notification_interval(unsigned char *mac_address, int interval);
```

Description

set_ip4_change_notification_interval changes the parameter of [Interval] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int interval

It is a value of Interval.

Return values

Each value returned by *set_ip4_change_notification_interval* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

-2

It returns -2 if [Protocol] in [Notify IPv4 Change] section is 2(TCP) or 3(UDP).

Examples

See also

set_ip4_change_notification_type,
set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_data_type,
set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

[Interval] in [Notify IPv4 Change] section is used when [Protocol] is 2(TCP) or 3(UDP).

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

5.7.19 set_ip4_change_notification_peer_port

```
int set_ip4_change_notification_peer_port(unsigned char *mac_address, int peer_port);
```

Description

set_ip4_change_notification_peer_port changes the parameter of [Port] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int interval



It is a TCP or UDP port number of remote host.

Return values

Each value returned by *set_ip4_change_notification_peer_port* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

set_ip4_change_notification_type,
set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_data_type,
set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

[Port] in [Notify IPv4 Change] section is used when [Protocol] is 2(TCP) or 3(UDP).

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

5.7.20 set_ip4_change_notification_peer_address

```
int set_ip4_change_notification_peer_address(unsigned char *mac_address,
                                             char *peer_address);
```

Description

set_ip4_change_notification_peer_address changes the parameter of [Host Name(dyndns)] or [Host Name(custom)] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *peer_address*

peer_address points to a character string containing an Host Name(dyndns) or Host Name(custom).

Return values

Each value returned by *set_ip4_change_notification_peer_address* is defined as follows:

1



If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

set_ip4_change_notification_type,
set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_data_type,
set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

If [Protocol] in [Notify IPv4 Change] section is 1(DDNS(dyndns.org)) then *peer_address* is used for [Host Name(dyndns)].

If [Protocol] in [Notify IPv4 Change] section is 2(TCP) or 3(UDP) then *peer_address* is used for [Host Name(custom)].

The length of [Host Name(dyndns)] or [Host Name(custom)] is 64 bytes include a NULL byte.

Please refer to product's user manual for more detail information about [Notify IPv4 Change].

5.7.21 set_ip4_change_notification_ddns_id

```
int set_ip4_change_notification_ddns_id(unsigned char *mac_address, char *ddns_id);
```

Description

set_ip4_change_notification_ddns_id changes the parameter of [DDNS ID] in [Notify IPv4 Change] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ddns_id*

ddns_id points to a character string containing an DDNS ID to use the DynDns service.

Return values

Each value returned by *set_ip4_change_notification_ddns_id* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.



Examples

See also

set_ip4_change_notification_type,
set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_data_type,
set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

[**DDNS ID**] in [**Notify IPv4 Change**] section is used when [**Protocol**] is 1(dyndns.org).

The length of [**DDNS ID**] is 32 bytes except a NULL byte.

Please refer to product's user manual for more detail information about [**Notify IPv4 Change**].

5.7.22 set_ip4_change_notification_ddns_pwd

```
int set_ip4_change_notification_ddns_pwd(unsigned char *mac_address,
                                         char *ddns_pwd);
```

Description

set_ip4_change_notification_ddns_pwd changes the parameter of [**DDNS Password**] in [**Notify IPv4 Change**] section.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *ddns_pwd*

ddns_id points to a character string containing an DDNS Password to use the DynDns service.

Return values

Each value returned by *set_ip4_change_notification_ddns_pwd* is defined as follows:

1

If the function succeeds, it returns 1.

-1

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

set_ip4_change_notification_type,

set_ip4_change_notification_data_type,

set_ip4_change_notification_interval,
set_ip4_change_notification_peer_address,
set_ip4_change_notification_ddns_pwd.

set_ip4_change_notification_peer_port,
set_ip4_change_notification_ddns_id,

Remarks

[**DDNS Password**] in [**Notify IPv4 Change**] section is used when [**Protocol**] is 1(dyndns.org).

The length of [**DDNS Password**] is 16 bytes except a NULL byte.

Please refer to product's user manual for more detail information about [**Notify IPv4 Change**].

6 Write parameters

6.1 EzTCP_Write

```
int EzTCP_Write(unsigned char *mac_address, char *password, int udp_port_number,
                int *error);
```

Description

This function write environmental parameters to a ezTCP designated by *mac_address*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *password*

password points to a character string containing a password. A password is required if a ezTCP is protected by passwords.

int udp_port_number

To designate a UDP port number for *EzTCP_Write*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_Write* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_PWD_LENGTH

The length of *password* is invalid. The length is 4 ~ 8 characters.

EZTCP_ERR_LOCAL_IP

The [Local IP Address] of ezTCP has unavailable value.

EZTCP_ERR_LOCAL_PORT_0

The value of [Local Port] is duplicated. The [Local Port] should have a unique number.



EZTCP_ERR_LOCAL_PORT_1

The number **23** can't be used for [**Local Port**] of a COM port if [**Telnet**] option is enabled.

EZTCP_ERR_LOCAL_PORT_2

In case of I/O products, if the method of making a Modbus/TCP connection is [**Passive Connection**] then the value of [**Local Port**] of I/O Controller can't be used for [**Local Port**] of a COM port.

EZTCP_ERR_LOCAL_PORT_3

In case of I/O products, if [**Web(HTTP)**] of I/O Controller is selected then the value of [Web(HTTP) port] can't be used for [Local Port] of a COM port.

EZTCP_ERR_LOCAL_PORT_4

The number **50,005** can't be used for [**Local Port**] of a COM port.

EZTCP_ERR_LOCAL_PORT_5

The number **50,006** can't be used for [**Local Port**] of a COM port.

EZTCP_ERR_LOCAL_PORT_6

The number **50,007** can't be used for [**Local Port**] of a COM port.

EZTCP_ERR_IO_SCRIPT

In case of I/O products, a MACRO text has a syntactic error.

EZTCP_ERR_IO_ADDR

In case of I/O products, [**Input Port Base Address**] or [**Output Port Base Address**] has unavailable number.

EZTCP_ERR_RES

The *EzTCP_Write* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_PWD

The password in ezTCP and a string pointed by *password* are not matched.

EZTCP_ERR_PRODUCT_MISMATCH

The product name of environmental parameters is not same as the product name designated by *mac_address*.

EZTCP_ERR_FLASH_NOSPACE

There is no enough memory space to save environmental variables.

EZTCP_ERR_UNKNOWN

The *EzTCP_Write* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
```

```

    unsigned char *mac_address = out_eztcp_info.mac_address;

    // change some variables by using set_XXXXX functions.

    int err = 0;
    int res = EzTCP_Write(mac_address, NULL, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        printf("The requested operation has been successfully completed!\r\n");
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("writing has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
            FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
            (LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
        }
    }
}

```

See also

EzTCP_Write, EzTCP_Initialize.

Remarks

6.2 EzTCP_Initialize

```

int EzTCP_Initialize(unsigned char *mac_address, char *password, int udp_port_number,
                    int *error);

```

Description

This function initialize environmental parameters in a ezTCP designated by *mac_address*.



Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *password*

password points to a character string containing a password. A password is required if a ezTCP is protected by passwords.

int udp_port_number

To designate a UDP port number for *EzTCP_Initialize*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_Initialize* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_PWD_LENGTH

The length of *password* is invalid. The length is 4 ~ 8 characters.

EZTCP_ERR_RES

The *EzTCP_Initialize* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_PWD

The password in ezTCP and a string pointed by *password* are not matched.

EZTCP_ERR_PRODUCT_MISMATCH

The product name of environmental parameters is not same as the product name designated by *mac_address*.

EZTCP_ERR_FLASH_NOSPACE

There is no enough memory space to save environmental variables.

EZTCP_ERR_UNKNOWN

The *EzTCP_Initialize* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples



```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    unsigned char *mac_address = out_eztcp_info.mac_address;

    int err = 0;
    int res = EzTCP_Initialize(mac_address, NULL, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        printf("The requested operation has been successfully completed!\r\n");
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("Initializing has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                (LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
        }
    }
}

```

See also

EzTCP_Write, EzTCP_Initialize.

Remarks

7 Management password

7.1 EzTCP_ChangePassword

```
int EzTCP_ChangePassword(unsigned char *mac_address, char *current_pwd,
                        char *new_pwd, int udp_port_number, int *error);
```

Description

This function saves, changes or deletes a password of a ezTCP designated by *mac_address*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *current_pwd*

current_pwd points to a character string containing a password. The character string should have the password of a ezTCP currently have. *current_pwd* is using to change or delete current password.

If a ezTCP does not have a password then *current_pwd* should be NULL or length of *current_pwd* should be 0.

*char *new_pwd*

new_pwd points to a character string containing a password to be written to a ezTCP.

If you want to delete current password then *new_pwd* should be NULL or the length of *new_pwd* should be 0.

int udp_port_number

To designate a UDP port number for *EzTCP_ChangePassword*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_ChangePassword* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_PWD_LENGTH

The length of *password* is invalid. The length is 4 ~ 8 characters.

EZTCP_ERR_RES

The *EzTCP_ChangePassword* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_PWD

The password in ezTCP and a string pointed by *password* are not matched.

EZTCP_ERR_PRODUCT_MISMATCH

The product name of environmental parameters is not same as the product name designated by *mac_address*.

EZTCP_ERR_FLASH_NOSPACE

There is no enough memory space to save environmental variables.

EZTCP_ERR_UNKNOWN

The *EzTCP_ChangePassword* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples

```
//-----
// example of setting new password
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char *new_pwd = "bbbb";
    unsigned char *mac_address = out_eztcp_info.mac_address;
    int err = 0;

    int res = EzTCP_ChangePassword(mac_address, NULL, new_pwd,
out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        printf("The requested operation has been successfully completed!\r\n");
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("changing the password has failed.[Error : %d]\r\n", err);
        }
    }
}
```

```

        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                (LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);

        }
    }

}

//-----
// example of changing current password
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char *cur_pwd = "bbbb"
    char *new_pwd = "aaaa";
    unsigned char *mac_address = out_eztcp_info.mac_address;
    int err = 0;

    int res = EzTCP_ChangePassword(mac_address, cur_pwd, new_pwd,
        out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        printf("The requested operation has been successfully completed!\r\n");
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("changing the password has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                (LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);

        }
    }
}

```

```

    }
}

//-----
// example of deleting current password
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char *cur_pwd = "aaaa"
    unsigned char *mac_address = out_eztcp_info.mac_address;
    int err = 0;

    int res = EzTCP_ChangePassword(mac_address, cur_pwd, NULL,
out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        printf("The requested operation has been successfully completed!\r\n");
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("Changing the password has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
        }
    }
}
}

```

See also

Remarks



8 Status monitoring

8.1 EzTCP_Status

```
int EzTCP_Status(unsigned char *mac_address, int udp_port_number, int *error);
```

Description

EzTCP_Status retrieves status information from a ezTCP. Status information is a plain text string. Each product may have different length of the string according to its firmware version or product's type.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int udp_port_number

To designate a UDP port number for *EzTCP_Status*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_Status* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_RES

The *EzTCP_Status* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_UNKNOWN

The *EzTCP_Status* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples



See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
 get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
 EzTCP_CloseTcpIp6.

Remarks

8.2 get_status_string_length

```
int get_status_string_length(void);
```

Description

get_status_string_length returns the length of status information string. The return value is valid after using *EzTCP_Status* function.

Parameters**Return values**

get_status_string returns the length of status information string.

Examples**See also**

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
 get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
 EzTCP_CloseTcpIp6.

Remarks

8.3 get_status_string

```
void get_status_string(char *buf);
```

Description

get_status_string retrieves status information string of a ezTCP. This function is valid after using *EzTCP_Status* function.

Parameters

*char *buf*

A pointer to char that status information string is saved.

Return values

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char *buf;
    int err = 0;
    int status_len = 0;

    int res = EzTCP_Status(out_eztcp_info.mac_address, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        status_len = get_status_string_length();
        if(status_len > 0)
        {
            buf = (char*)malloc(status_len + 1);
            memset(buf, 0x00, status_len + 1);
            get_status_string(buf);
            printf("[Status]\\r\\n%s\\r\\n", buf);
            free(buf);
        }
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("The requested operation has failed.[Error : %d]\\r\\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
                FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
                (LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\\r\\n", err, (char*)lpMsgBuf);
        }
    }
}
```

```

    }
}

```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count, get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4, EzTCP_CloseTcpIp6.

Remarks

buf should have enough memory space at least amount of the returned value of *get_status_string_length*.

8.4 get_tcpip4_session_count

```
int get_tcpip4_session_count(void);
```

Description

get_tcpip4_session_count returns the total number of TCP/IP version4 session that a ezTCP can have. This function is valid after using *EzTCP_Status* function.

Parameters

Return values

get_tcpip4_session_count returns the total number of TCP/IP version4 session.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[256];
    int err = 0;
    int idx;
    int tcpip4_count;

    int res = EzTCP_Status(out_eztcp_info.mac_address, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        tcpip4_count = get_tcpip4_session_count();
    }
}

```

```

        for(idx = 0;idx < tcpip4_count;idx++)
        {
            if(get_tcpip4_session_info(idx, buf) == 1)
            {
                memset(buf, 0x00, 256);
                printf("[TCP/IP4] SESSION #%d\r\n%s\r\n", idx, buf);
            }
        }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("The requested operation has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
            FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
            (LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
        }
    }
}

```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
 get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
 EzTCP_CloseTcpIp6.

Remarks

8.5 get_tcpip6_session_count

```
int get_tcpip6_session_count(void);
```

Description

get_tcpip6_session_count returns the total number of TCP/IP version6 session that a ezTCP can have.
 This function is valid after using *EzTCP_Status* function.

Parameters

Return values

get_tcpip6_session_count returns the total number of TCP/IP version6 session.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[256];
    int err = 0;
    int idx;
    int tcpip6_count;

    int res = EzTCP_Status(out_eztcp_info.mac_address, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        tcpip6_count = get_tcpip6_session_count();

        for(idx = 0; idx < tcpip6_count; idx++)
        {
            if(get_tcpip6_session_info(idx, buf) == 1)
            {
                memset(buf, 0x00, 256);
                printf("[TCP/IP6] SESSION #%d\r\n%s\r\n", idx, buf);
            }
        }
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("The requested operation has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;
```

```

        FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpmgBuf, 0, NULL);

        printf("Error[%d] %s\r\n", err, (char*)lpmgBuf);

    }

}

}

```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
EzTCP_CloseTcpIp6.

Remarks

8.6 get_tcpip4_session_info

```
int get_tcpip4_session_info(int session_number, char *out_session_info);
```

Description

get_tcpip4_session_info retrieves a connection status of TCP/IP version4 session designated by *session_number*. This function is valid after using *EzTCP_Status* function.

Parameters

int session_number

The ordinal number of TCP/IP version4 session. It is started from 0 and less than the returned value of *get_tcpip4_session_count*.

*char *out_session_info*

A pointer to char that a connection status of TCP/IP version4 session is saved. *out_session_info* should have enough memory space more than 256-byte.

Return values

If no error occurs, *get_tcpip4_session_info* returns 1. Otherwise, 0 is returned.

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[256];

```

```

int err = 0;
int idx;
int tcpip4_count;

int res = EzTCP_Status(out_eztcp_info.mac_address, out_eztcp_info.search_port, &err);

if(res == EZTCP_SUCCESS)
{
    tcpip4_count = get_tcpip4_session_count();
    for(idx = 0; idx < tcpip4_count; idx++)
    {
        if(get_tcpip4_session_info(idx, buf) == 1)
        {
            memset(buf, 0x00, 256);
            printf("[TCP/IP4] SESSION #%d\r\n%s\r\n", idx, buf);
        }
    }
}
else if(res == EZTCP_ERR)
{
    if(err < 0)
    {
        printf("The requested operation has failed.[Error : %d]\r\n", err);
    }
    else if(err > 0)
    {
        LPVOID lpMsgBuf;

        FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
        FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
        (LPSTR)&lpMsgBuf, 0, NULL);

        printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
    }
}
}

```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
EzTCP_CloseTcpIp6.

Remarks

out_session_info should have enough memory space more than 256-byte.



8.7 get_tcpip6_session_info

```
int get_tcpip6_session_info(int session_number, char *out_session_info);
```

Description

get_tcpip6_session_info retrieves a connection status of TCP/IP version6 session designated by *session_number*. This function is valid after using *EzTCP_Status* function.

Parameters

int session_number

The ordinal number of TCP/IP version6 session. It is started from 0 and less than the returned value of *get_tcpip6_session_count*.

*char *out_session_info*

A pointer to char that a connection status of TCP/IP version6 session is saved. *out_session_info* should have enough memory space more than 256-byte.

Return values

If no error occurs, *get_tcpip6_session_info* returns 1. Otherwise, 0 is returned.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[256];
    int err = 0;
    int idx;
    int tcpip6_count;

    int res = EzTCP_Status(out_eztcp_info.mac_address, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        tcpip6_count = get_tcpip6_session_count();

        for(idx = 0; idx < tcpip6_count; idx++)
        {
            if(get_tcpip6_session_info(idx, buf) == 1)
            {
                memset(buf, 0x00, 256);
            }
        }
    }
}
```

```

        printf("[TCP/IP6] SESSION #d\r\n%s\r\n", idx, buf);
    }
}
else if(res == EZTCP_ERR)
{
    if(err < 0)
    {
        printf("The requested operation has failed.[Error : %d]\r\n", err);
    }
    else if(err > 0)
    {
        LPVOID lpMsgBuf;

        FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpMsgBuf, 0, NULL);

        printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
    }
}
}
}

```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
EzTCP_CloseTcpIp6.

Remarks

out_session_info should have enough memory space more than 256-byte.

8.8 EzTCP_CloseTcpIp4

```
int EzTCP_CloseTcpIp4(unsigned char *mac_address, int session_number,
char *password, int *error);
```

Description

EzTCP_CloseTcpIp4 changes a connection status of TCP/IP version4 session into **CLOSED**, if the connection status is **ESTABLISHED**. The TCP/IP version4 session is designated by *session_number*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



int session_number

The ordinal number of TCP/IP version4 session. It is started from 0 and less than the returned value of *get_tcpip4_session_count*.

*char *password*

password points to a character string containing a password. A password is required if a ezTCP is protected by passwords.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_CloseTcpIp4* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_PWD_LENGTH

The length of *password* is invalid. The length is 4 ~ 8 characters.

EZTCP_ERR_RES

The *EzTCP_CloseTcpIp4* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_PWD

The password in ezTCP and a string pointed by *password* are not matched.

EZTCP_ERR_UNKNOWN

The *EzTCP_CloseTcpIp4* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[256];
    int err = 0;
    int idx;
```

```

int tcpip4_count;
unsigned char *mac_address = out_eztcp_info.mac_address;

int res = EzTCP_Status(mac_address, out_eztcp_info.search_port, &err);

if(res == EZTCP_SUCCESS)
{
    tcpip4_count = get_tcpip4_session_count();

    if(tcpip4_count <= 0)
        return;

    err = 0;

    res = EzTCP_CloseTcpIp4(mac_address, 0, NULL, &err);
    if(res == EZTCP_SUCCESS)
    {
        printf("Success.\r\n");
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("The requested operation has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;
            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
                FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err,
                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPSTR)&lpMsgBuf, 0, NULL);
            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
        }
    }
}
}

```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
 get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
 EzTCP_CloseTcpIp6.

Remarks



8.9 EzTCP_CloseTcpIp6

```
int EzTCP_CloseTcpIp6(unsigned char *mac_address, int session_number,
                      char *password, int *error);
```

Description

EzTCP_CloseTcpIp6 changes a connection status of TCP/IP version6 session into **CLOSED**, if the connection status is **ESTABLISHED**. The TCP/IP version6 session is designated by *session_number*.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int session_number

The ordinal number of TCP/IP version6 session. It is started from 0 and less than the returned value of *get_tcpip6_session_count*.

*char *password*

password points to a character string containing a password. A password is required if a ezTCP is protected by passwords.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_CloseTcpIp6* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_PWD_LENGTH

The length of *password* is invalid. The length is 4 ~ 8 characters.

EZTCP_ERR_RES

The *EzTCP_CloseTcpIp6* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_PWD

The password in ezTCP and a string pointed by *password* are not matched.

EZTCP_ERR_UNKNOWN

The *EzTCP_CloseTcpIp6* function receives unknown error code from a ezTCP designated by

mac_address.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[256];
    int err = 0;
    int idx;
    int tcpip6_count;
    unsigned char *mac_address = out_eztcp_info.mac_address;

    int res = EzTCP_Status(mac_address, out_eztcp_info.search_port, &err);

    if(res == EZTCP_SUCCESS)
    {
        tcpip6_count = get_tcpip6_session_count();

        if(tcpip6_count <= 0)
            return;

        err = 0;

        res = EZTCP_CloseTcpIp6(mac_address, 0, NULL, &err);
        if(res == EZTCP_SUCCESS)
        {
            printf("Success.\r\n");
        }
        else if(res == EZTCP_ERR)
        {
            if(err < 0)
            {
                printf("The requested operation has failed.[Error : %d]\r\n", err);
            }
            else if(err > 0)
            {
                LPVOID lpMsgBuf;
```

```
        FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |  
        FORMAT_MESSAGE_FROM_SYSTEM | FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err,  
        MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), (LPSTR)&lpmgBuf, 0, NULL);  
        printf("Error[%d] %s\r\n", err, (char*)lpmgBuf);  
    }  
}  
}
```

See also

EzTCP_Status, get_status_string_length, get_status_string, get_tcpip4_session_count,
get_tcpip6_session_count, get_tcpip4_session_info, get_tcpip6_session_info, EzTCP_CloseTcpIp4,
EzTCP_CloseTcpIp6.

Remarks



9 Certificates

9.1 EzTCP_ReadCert

```
int EzTCP_ReadCert(unsigned char *mac_address, int udp_port_number, int *error);
```

Description

EzTCP_ReadCert retrieves the certificate information stored in a ezTCP. The certificate information is a plain text string, and the length may vary depending on the certificate.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int udp_port_number

To designate a UDP port number for *EzTCP_ReadCert*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_ReadCert* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_RES

The *EzTCP_Status* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_UNKNOWN

The *EzTCP_Status* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples



See also

`is_certificates`, `EzTCP_ReadCert`, `get_certificates_string_length`, `get_certificates_string`,
`EzTCP_WriteSelfSignedCertificates`, `EzTCP_WriteCertificates`.

Remarks

9.2 `get_certificates_string_length`

```
int get_certificates_string_length(void);
```

Description

get_certificates_string_length returns the length of the certificate information string. The return value is valid after using *EzTCP_ReadCert* function.

Parameters**Return values**

get_certificates_string_length returns the length of the certificate string.

Examples**See also**

`is_certificates`, `EzTCP_ReadCert`, `get_certificates_string_length`, `get_certificates_string`,
`EzTCP_WriteSelfSignedCertificates`, `EzTCP_WriteCertificates`.

Remarks

9.3 `get_certificates_string`

```
void get_certificates_string(char *buf);
```

Description

get_certificates_string retrieves the certificate information string of a ezTCP. This function is valid after using *EzTCP_ReadCert* function.

Parameters

*char *buf*

A pointer to char that certificates information string is saved.

Return values

Examples

```

struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char *buf;
    int err = 0;
    int certificates_len = 0;

    int res = EzTCP_ReadCert(out_eztcp_info.mac_address, out_eztcp_info.search_port,
&err);

    if(res == EZTCP_SUCCESS)
    {
        certificates_len = get_certificates_string_length();
        if(certificates_len > 0)
        {
            buf = (char*)malloc(certificates_len + 1);
            memset(buf, 0x00, certificates_len + 1);
            get_certificates_string(buf);
            printf("[Certificates]\r\n%s\r\n", buf);
            free(buf);
        }
    }
    else if(res == EZTCP_ERR)
    {
        if(err < 0)
        {
            printf("The requested operation has failed.[Error : %d]\r\n", err);
        }
        else if(err > 0)
        {
            LPVOID lpMsgBuf;

            FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER | FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS, NULL, err, MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
(LPSTR)&lpMsgBuf, 0, NULL);

            printf("Error[%d] %s\r\n", err, (char*)lpMsgBuf);
        }
    }
}
}

```

See also

is_certificates, EzTCP_ReadCert, get_certificates_string_length, get_certificates_string,
EzTCP_WriteSelfSignedCertificates, EzTCP_WriteCertificates.

Remarks

9.4 EzTCP_WriteSelfSignedCertificates

```
int EzTCP_WriteSelfSignedCertificates(unsigned char *mac_address, int rsa_key_length,
char *country_code, char *state_name, char *locality_name, char *organization_name,
char *organization_unit_name, char *common_name, char *email, int udp_port_number,
int *error);
```

Description

This function write a self-signed certificate to a ezTCP designated by mac_address.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int rsa_key_length

RSA key length. Valid values are 1024, 1536, and 2048.

*char *country_code*

Country name in two-letter code.

*char *state_name*

State of province name.

*char *locality_name*

Locality name.

*char *organization_name*

Organization name. Use something like your company name.

*char *organization_unit_name*

Organizational unit name. Use something like the department name.

*char *common_name*

Common name. Use something like the server address.

*char *email*

Email address.

int udp_port_number

To designate a UDP port number for *EzTCP_Status*. If set to 0, the basic port number (50005 or 50007) will be used.



*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_WriteSelfSignedCertificates* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_RES

The *EzTCP_Status* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_UNKNOWN

The *EzTCP_Status* function receives unknown error code from a ezTCP designated by *mac_address*.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples

See also

is_certificates, *EzTCP_ReadCert*, *get_certificates_string_length*, *get_certificates_string*,
EzTCP_WriteSelfSignedCertificates, *EzTCP_WriteCertificates*.

Remarks

9.5 EzTCP_WriteCertificates

```
int EzTCP_WriteCertificates(unsigned char *mac_address, char *certificates_filepath,
char *certificates_password, char *rsa_key_filepath, char *rsa_key_password, int
udp_port_number, int *error);
```

Description

This function write a signed certificate from certification authorities to a ezTCP designated by *mac_address*.

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *certificates_filepath*

The path where the certificate file is located.

*char *certificates_password*

If a password is set in the certificate file, enter that password.

*char *ras_key_filepath*

The path where the key file is located if the certificate file does not include the key together. This parameter is not required..

*char *rsa_key_password*

If a password is set in the key file, enter that password.

int udp_port_number

To designate a UDP port number for *EzTCP_Status*. If set to 0, the basic port number (50005 or 50007) will be used.

*int *error*

If the function fails, an error code will be saved.

Return values

Each value returned by *EzTCP_WriteCertificates* is defined as follows:

EZTCP_SUCCESS

If no error occurs, this function returns *EZTCP_SUCCESS*.

EZTCP_ERR

If an error occurs, this function returns *EZTCP_ERR* and saves a specific error code to *error*.

If the value of *error* is less than 0, it is defined as follows:

EZTCP_ERR_NO_INFO

The ezTCP with a *mac_address* is not found.

EZTCP_ERR_RES

The *EzTCP_Status* function didn't receive a response packet from a ezTCP.

EZTCP_ERR_UNKNOWN

The *EzTCP_Status* function receives unknown error code from a ezTCP designated by *mac_address*.

EZTCP_ERR_FOPEN

The file cannot be found or read.

EZTCP_ERR_READ_CERT

If the certificate cannot be resolved.

EZTCP_ERR_NO_RSA_KEY

If the RSA key cannot be found.

EZTCP_ERR_NO_CERT

If the certificate cannot be found.

EZTCP_ERR_READ_RSA_KEY

If the key cannot be resolved.

If the value of *error* is greater than 0, it is one of system errors came from user's PC.

Examples

See also

is_certificates, EzTCP_ReadCert, get_certificates_string_length, get_certificates_string,
EzTCP_WriteSelfSignedCertificates, EzTCP_WriteCertificates.

Remarks



10 Additional functions

10.1 get_version

```
int get_version(unsigned char *mac_address, char *out_version);
```

Description

get_version retrieves a firmware information of ezTCP.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

*char *out_version*

A pointer to char that firmware information string is saved.

Return values

If no error occurs, *get_tcpip4_session_info* returns 1.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

```
struct eztcp_info out_eztcp_info;
int ret = get_eztcp_info_by_index(0, &out_eztcp_info);

if(ret == EZTCP_SUCCESS)
{
    char buf[8];
    memset(buf, 0x00, 8);
    if(get_version(out_eztcp_info.mac_address, buf) == 1)
    {
        printf("Firmware version : %s\r\n", buf);
    }
}
```

See also

Remarks

The firmware version text consisted of major number, minor number and revision character. For example, *out_version* may have "1.1A". First '1' is a major number, second '1' is a minor number and last 'A' is a revision character.

10.2 is_ip6

```
int is_ip6(unsigned char *mac_address);
```

Description

is_ip6 retrieves whether TCP/IP version6 function is available or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_ip6 returns 1 if the ezTCP designated by *mac_address* can use **TCP/IP version6 function** NOW. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_ip6* may be changed according to another environmental parameters.

10.3 is_uart_rs232

```
int is_uart_rs232(unsigned char *mac_address, int uart_index);
```

Description

is_uart_rs232 retrieves whether a COM port designated by *uart_index* can use RS-232 or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_rs232 returns 1 if a COM port designated by *uart_index* can use **RS-232** NOW. Otherwise it

returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.4 is_uart_rs485

```
int is_uart_rs485(unsigned char *mac_address, int uart_index);
```

Description

is_uart_rs485 retrieves whether a COM port designated by *uart_index* can use RS-485 or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_rs485 returns 1 if a COM port designated by *uart_index* can use **RS-485** NOW. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_rs485* may be changed according to another environmental parameters.

10.5 is_uart_rs422

```
int is_uart_rs422(unsigned char *mac_address, int uart_index);
```

Description

is_uart_rs422 retrieves whether a COM port designated by *uart_index* can use RS-422 or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_rs422 returns 1 if a COM port designated by *uart_index* can use **RS-422** NOW. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_rs422* may be changed according to another environmental parameters.

10.6 is_uart_ttl

```
int is_uart_ttl(unsigned char *mac_address);
```

Description

is_uart_ttl retrieves whether a ezTCP can use **TTL level signals** for it's COM ports or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_ttl returns 1 if the ezTCP designated by *mac_address* can use **TTL level signal** for it's COM ports. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples



See also**Remarks****10.7 is_uart_8_databit_only**

```
int is_uart_8_databit_only(unsigned char *mac_address);
```

Description

is_uart_8_databit_only retrieves whether COM ports in ezTCP supports 8-bit only for **[Data Bits]** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_8_databit_only returns 1 if the ezTCP designated by *mac_address* supports 8-bit only for **[Data Bits]**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks****10.8 is_uart_7_and_8_databit_only**

```
int is_uart_7_and_8_databit_only(unsigned char *mac_address);
```

Description

is_uart_7_and_8_databit_only retrieves whether COM ports in ezTCP supports 7 and 8-bit only for **[Data Bits]** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.



Return values

is_uart_7_and_8_databit_only returns 1 if the ezTCP designated by *mac_address* supports 7 and 8-bit only for **[Data Bits]**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

10.9 is_uart_one5_stopbit

```
int is_uart_one5_stopbit(unsigned char *mac_address);
```

Description

is_uart_one5_stopbit retrieves whether COM ports in ezTCP supports 1.5-bit for **[Stop Bit]** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_one5_stopbit returns 1 if the ezTCP designated by *mac_address* supports 1.5-bit for **[Stop Bit]**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

10.10 is_uart_dtrdsr

```
int is_uart_dtrdsr(unsigned char *mac_address);
```

Description

is_uart_dtrdsr retrieves whether COM ports in ezTCP supports DTR/DSR flow control or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_dtrdsr returns 1 if the ezTCP designated by *mac_address* supports DTR/DSR flow control. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.11 is_uart_xonxoff

```
int is_uart_xonxoff(unsigned char *mac_address);
```

Description

is_uart_xonxoff retrieves whether COM ports in ezTCP supports XON/XOFF flow control or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_xonxoff returns 1 if the ezTCP designated by *mac_address* supports XON/XOFF flow control. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks



10.12 is_uart_tx_delay

```
int is_uart_tx_delay(unsigned char *mac_address);
```

Description

is_uart_tx_delay retrieves whether COM ports in ezTCP supports **[TX Interval] function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_tx_delay returns 1 if the ezTCP designated by *mac_address* supports **[TX Interval] function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.13 is_uart_data_frame_interval

```
int is_uart_data_frame_interval(unsigned char *mac_address);
```

Description

is_uart_data_frame_interval retrieves whether COM ports in ezTCP supports **[Data Frame Interval(10ms)] function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_data_frame_interval returns 1 if the ezTCP designated by *mac_address* supports **[Data Frame Interval(10ms)] function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.14 `is_uart_separator`

```
int is_uart_separator(unsigned char *mac_address);
```

Description

is_uart_separator retrieves whether COM ports in ezTCP supports **Separator function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_separator returns 1 if the ezTCP designated by *mac_address* supports **Separator function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.15 `is_uart_tcp_nodelay`

```
int is_uart_tcp_nodelay(unsigned char *mac_address, int uart_index);
```

Description

is_uart_tcp_nodelay retrieves whether COM ports in ezTCP supports **[Disable TCP Transmission Delay] function** or not.

Parameters

*unsigned char *mac_address*



MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_tcp_nodelay returns 1 if the ezTCP designated by *mac_address* supports [**Disable TCP Transmission Delay**] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_tcp_nodelay* may be changed according to another environmental parameters.

10.16 is_uart_rfc2217

```
int is_uart_rfc2217(unsigned char *mac_address, int uart_index);
```

Description

is_uart_rfc2217 retrieves whether COM ports in ezTCP supports [**Telnet COM Port Control(RFC2217)**] **function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_rfc2217 returns 1 if the ezTCP designated by *mac_address* supports [**Telnet COM Port Control(RFC2217)**] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_rfc2217* may be changed according to another environmental parameters.

10.17 is_uart_protocol

```
int is_uart_protocol(unsigned char *mac_address);
```

Description

is_uart_protocol retrieves whether COM ports in ezTCP supports [**Protocol**] **function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_uart_protocol returns 1 if the ezTCP designated by *mac_address* supports [**Protocol**] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

Currently, this function only supports CSE-T16, CSE-T32 and CSE-T48. A supported products may be added in future. Please refer to product's user manual for more detail information.

10.18 is_uart_tcp_server

```
int is_uart_tcp_server(unsigned char *mac_address, int uart_index);
```

Description

is_uart_tcp_server retrieves whether COM ports in ezTCP supports **T2S - TCP Server mode** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values



is_uart_tcp_server returns 1 if a COM port designated by *uart_index* can work as a **TCP Server**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_tcp_server* may be changed according to another environmental parameters.

10.19 is_uart_at_command

```
int is_uart_at_command(unsigned char *mac_address, int uart_index);
```

Description

is_uart_at_command retrieves whether COM ports in ezTCP supports **ATC - AT Command mode** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_at_command returns 1 if a COM port designated by *uart_index* can use **AT Command mode**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_at_command* may be changed according to another environmental parameters.

10.20 is_uart_tcp_client

```
int is_uart_tcp_client(unsigned char *mac_address, int uart_index);
```

Description

is_uart_tcp_client retrieves whether COM ports in ezTCP supports **COD - TCP Client mode** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_tcp_client returns 1 if a COM port designated by *uart_index* can work as a **TCP Client**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_uart_tcp_client* may be changed according to another environmental parameters.

10.21 is_uart_udp

```
int is_uart_udp(unsigned char *mac_address, int uart_index);
```

Description

is_uart_udp retrieves whether COM ports in ezTCP supports **U2S - UDP mode** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_udp returns 1 if a COM port designated by *uart_index* can use **UDP**. Otherwise it returns 0.
If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

The return value of *is_uart_udp* may be changed according to another environmental parameters.

10.22 is_uart_serial_modbus

```
int is_uart_serial_modbus(unsigned char *mac_address, int uart_index);
```

Description

is_uart_serial_modbus retrieves whether COM ports in ezTCP supports **Serial Modbus/TCP mode** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

int uart_index

The ordinal number of COM ports. It starts from 0.

Return values

is_uart_serial_modbus returns 1 if a COM port designated by *uart_index* can use **Serial Modbus/TCP mode**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

The return value of *is_uart_serial_modbus* may be changed according to another environmental parameters.

10.23 is_csc_hr2

```
int is_csc_hr2(unsigned char *mac_address);
```

Description

is_csc_hr2 retrieves whether a ezTCP is a CSC-HR2 or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_csc_hr2 returns 1 if a ezTCP designated by *mac_address* is a **CSC-HR2**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.24 is_io

```
int is_io(unsigned char *mac_address);
```

Description

is_io retrieves whether a ezTCP is a I/O Controller or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_io returns 1 if a ezTCP designated by *mac_address* is a **I/O Controller** Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples



See also**Remarks****10.25 is_io_output_automatic_initialize**

```
int is_io_output_automatic_initialize(unsigned char *mac_address);
```

Description

is_io_output_automatic_initialize retrieves whether an I/O Controller supports **[Initialize the output port state(The output port will be changed to the[Initial State] when Modbus/TCP is disconnected.)]** function or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_io_output_automatic_initialize returns 1 if the ezTCP designated by *mac_address* supports **[Initialize the output port state(The output port will be changed to the[Initial State] when Modbus/TCP is disconnected.)]** function. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks****10.26 is_event_notification**

```
int is_event_notification(unsigned char *mac_address);
```

Description

is_event_notification retrieves whether a ezTCP supports **[Notify Input or Output Port Change(Email)]** function or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_event_notification returns 1 if the ezTCP designated by *mac_address* supports [**Notify Input or Output Port Change(Email)**] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.27 is_wlan

```
int is_wlan(unsigned char *mac_address);
```

Description

is_wlan retrieves whether a ezTCP is a Wireless LAN product or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_wlan returns 1 if the ezTCP designated by *mac_address* is a Wireless LAN product. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks



10.28 is_wlan_soft_ap

```
int is_wlan_soft_ap(unsigned char *mac_address);
```

Description

is_wlan_soft_ap retrieves whether a ezTCP(Wireless LAN product) supports **Soft AP function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_wlan_soft_ap returns 1 if a ezTCP support(Wireless LAN product) **Soft AP** function. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.29 is_wlan_antenna

```
int is_wlan_antenna(unsigned char *mac_address);
```

Description

is_wlan_antenna retrieves whether a ezTCP(Wireless LAN product) supports **[Antenna] option** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_wlan_antenna returns 1 if a ezTCP(Wireless LAN product) support **[Antenna]** option. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.30 is_wlan_phy_mode

```
int is_wlan_phy_mode(unsigned char *mac_address);
```

Description

is_wlan_phy_mode retrieves whether a ezTCP(Wireless LAN product) supports [**Advanced Settings**] in WLAN section or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_wlan_phy_mode returns 1 if a ezTCP(Wireless LAN product) supports [**Advanced Settings**] in WLAN section. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.31 is_wlan_background_scan

```
int is_wlan_background_scan(unsigned char *mac_address);
```

Description

is_wlan_background_scan retrieves whether a ezTCP(Wireless LAN product) supports [**Background Scan**] option in the wireless LAN [**Advanced Settings**] or not.

Parameters



*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_wlan_background_scan returns 1 if a ezTCP(Wireless LAN product) supports [**Background Scan**] option in the wireless LAN [**Advanced Settings**]. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.32 is_wlan_rsn

```
int is_wlan_rsn(unsigned char *mac_address);
```

Description

is_wlan_rsn retrieves whether a ezTCP(Wireless LAN product) supports **RSN(Robust Security Network)** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_wlan_rsn returns 1 if a ezTCP(Wireless LAN product) supports **RSN(Robust Security Network)**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

A Wireless LAN product that support **RSN(Robust Security Network)** automatically choose encryption and authentication method without related environmental parameters. So, we provide different functions for ezTCP according to whether it supports **RSN(Robust Security Network)** or not.

10.33 is_wlan_authentication_type

```
int is_wlan_authentication_type(unsigned char *mac_address);
```

Description

In cas of is_wlan_rsn returns 0, is_wlan_authentication_type retrieves available method of WPA [Authentication / Encryption].

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

If is_wlan_authentication_type return 0 then A Wireless LAN product can use WPA PSK – TKIP or WPA2 PSK – AES for WPA [Authentication / Encryption].

If is_wlan_authentication_type return 1 then A Wireless LAN product can use WPA PSK – TKIP, WPA PSK – AES, WPA PSK – TKIP/AES, WPA2 PSK – TKIP, WPA2 PSK – AES or WPA2 PSK – TKIP/AES for WPA [Authentication / Encryption].

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.34 is_wlan_wpa_enterprise

```
int is_wlan_wpa_enterprise(unsigned char *mac_address);
```

Description

is_wlan_wpa_enterprise retrieves whether a ezTCP(Wireless LAN product) supports [802.1X] function or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

`is_wlan_wpa_enterprise` returns 1 if a Wireless LAN product supports [802.1X] in the wireless LAN [Security Settings]. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**10.35 `is_send_mac_address`

```
int is_send_mac_address(unsigned char *mac_address);
```

Description

is_send_mac_address retrieves whether a ezTCP supports [Send MAC Address] function or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_send_mac_address returns 1 if the ezTCP designated by *mac_address* supports [Send MAC Address] function. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

The return value of *is_send_mac_address* may be changed according to another environmental parameters.

10.36 `is_ssl`

```
int is_ssl(unsigned char *mac_address);
```

Description

is_ssl retrieves whether a ezTCP supports [SSL] **function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_ssl returns 1 if the ezTCP designated by *mac_address* supports [SSL] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

The return value of *is_ssl* may be changed according to another environmental parameters.

10.37 is_ssh

```
int is_ssh(unsigned char *mac_address);
```

Description

is_ssh retrieves whether a ezTCP supports [SSH] **function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_ssh returns 1 if the ezTCP designated by *mac_address* supports [SSH] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also**

Remarks

The return value of *is_ssh* may be changed according to another environmental parameters.

10.38 is_remote_debug

```
int is_remote_debug(unsigned char *mac_address);
```

Description

is_remote_debug retrieves whether a ezTCP supports **[Debugging Message] function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_remote_debug returns 1 if the ezTCP designated by *mac_address* supports **[Debugging Message] function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples**See also****Remarks**

10.39 is_tcp_multi_connection

```
int is_tcp_multi_connection(unsigned char *mac_address);
```

Description

is_tcp_multi_connection retrieves whether a ezTCP supports **[Multiple Connection] function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_tcp_multi_connection returns 1 if the ezTCP designated by *mac_address* supports [**Multiple Connection**] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

The return value of *is_tcp_multi_connection* may be changed according to another environmental parameters.

10.40 is_power_management

```
int is_power_management(unsigned char *mac_address);
```

Description

is_power_management retrieves whether a ezTCP supports [**Power Management**] **function** or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_power_management returns 1 if the ezTCP designated by *mac_address* supports [**Power Management**] **function**. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

10.41 is_password

```
int is_password(unsigned char *mac_address);
```

Description



is_password retrieves whether a ezTCP is protected by passwords or not.

Parameters

*unsigned char *mac_address*

MAC address of the searched ezTCP by *EzTCP_Search* or *EzTCP_Read*.

Return values

is_password returns 1 if a ezTCP has a password. Otherwise it returns 0.

If the ezTCP with a *mac_address* is not found, then it returns -1.

Examples

See also

Remarks

11 Exemption from Liability

11.1 Exemption from Liability

11.1.1 English

In no event shall Sollae Systems Co., Ltd. and its distributors be liable for any damages whatsoever (including, without limitation, damages for loss of profit, operating cost for commercial interruption, loss of information, or any other financial loss) from the use or inability to use the ezTCP even if Sollae Systems Co., Ltd. or its distributors have been informed of such damages.

The ezTCP is not designed and not authorized for use in military applications, in nuclear applications, in airport applications or for use in applications involving explosives, or in medical applications, or for use in security alarm, or for use in a fire alarm, or in applications involving elevators, or in embedded applications in vehicles such as but not limited to cars, planes, trucks, boats, aircraft, helicopters, etc..

In the same way, the ezTCP is not designed, or intended, or authorized to test, develop, or be built into applications where failure could create a dangerous situation that may result in financial losses, damage to property, personal injury, or the death of people or animals. If you use the ezTCP voluntarily or involuntarily for such unauthorized applications, you agree to subtract Sollae Systems Co., Ltd. and its distributors from all liability for any claim for compensation.

Sollae Systems Co., Ltd. and its distributors entire liability and your exclusive remedy shall be Sollae Systems Co., Ltd. and its distributors option for the return of the price paid for, or repair, or replacement of the ezTCP.

Sollae Systems Co., Ltd. and its distributors disclaim all other warranties, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose, with respect to the ezTCP including accompanying written material, hardware and firmware.

11.1.2 French

- **Documentation**

La documentation du boîtier ezTCP est conçue avec la plus grande attention. Tous les efforts ont été mis en œuvre pour éviter les anomalies. Toutefois, nous ne pouvons garantir que cette documentation soit à 100% exempt de toute erreur. Les informations présentes dans cette documentation sont données à titre indicatif. Les caractéristiques techniques peuvent changer à tout moment sans aucun préavis dans le but d'améliorer la qualité et les possibilités des produits.

- **Copyright et appellations commerciales**

Toutes les marques, les procédés, les références et les appellations commerciales des produits cités dans la documentation appartiennent à leur propriétaire et Fabricant respectif.

- **Conditions d'utilisations et limite de responsabilité**

En aucun cas Sollae Systems Co., Ltd. ou un de ses distributeurs ne pourra être tenu responsable de dommages quels qu'ils soient (intégrant, mais sans limitation, les dommages pour perte de bénéfice commercial, interruption d'exploitation commerciale, perte d'informations et de données à caractère



commercial ou de toute autre perte financière) provenant de l'utilisation ou de l'incapacité à pouvoir utiliser le boîtier ezTCP, même si Sollae Systems Co., Ltd. ou un de ses distributeurs a été informé de la possibilité de tels dommages.

Le boîtier ezTCP est exclusivement prévu pour un usage en intérieur, dans un environnement sec, tempéré (+10 °C à +40°C) et non poussiéreux. Le boîtier ezTCP n'est pas prévu, ni autorisé pour être utilisé en extérieur, ni de façon embarquée dans des engins mobiles de quelque nature que ce soit (voiture, camion, train, avion, etc...), ni en milieu explosif, ni dans des enceintes nucléaires, ni dans des ascenseurs, ni dans des aéroports, ni dans des enceintes hospitaliers, ni pour des applications à caractère médical, ni dans des dispositifs de détection et d'alerte anti-intrusion, ni dans des dispositifs de détection et d'alerte anti-incendie, ni dans des dispositifs d'alarme GTC, ni pour des applications militaires.

De même, le boîtier ezTCP n'est pas conçu, ni destiné, ni autorisé pour expérimenter, développer ou être intégré au sein d'applications dans lesquelles une défaillance de celui-ci pourrait créer une situation dangereuse pouvant entraîner des pertes financières, des dégâts matériel, des blessures corporelles ou la mort de personnes ou d'animaux. Si vous utilisez le boîtier ezTCP volontairement ou involontairement pour de telles applications non autorisées, vous vous engagez à soustraire Sollae Systems Co., Ltd. et ses distributeurs de toute responsabilité et de toute demande de dédommagement.

En cas de litige, l'entière responsabilité de Sollae Systems Co., Ltd. et de ses distributeurs vis-à-vis de votre recours durant la période de garantie se limitera exclusivement selon le choix de Sollae Systems Co., Ltd. et de ses distributeurs au remboursement de votre produit ou de sa réparation ou de son échange. Sollae Systems Co., Ltd. et ses distributeurs démentent toutes autres garanties, exprimées ou implicites.

Tous les boîtiers ezTCP sont testés avant expédition. Toute utilisation en dehors des spécifications et limites indiquées dans cette documentation ainsi que les court-circuit, les chocs, les utilisations non autorisées, pourront affecter la fiabilité, créer des dysfonctionnements et/ou la destruction du boîtier ezTCP sans que la responsabilité de Sollae Systems Co., Ltd. et de ses distributeurs ne puissent être mise en cause, ni que le boîtier ezTCP puisse être échangé au titre de la garantie.

● Rappel sur l'évacuation des équipements électroniques usagés

Le symbole de la poubelle barré présent sur le boîtier ezTCP indique que vous ne pouvez pas vous débarrasser de ce dernier de la même façon que vos déchets courants. Au contraire, vous êtes responsable de l'évacuation du boîtier ezTCP lorsqu'il arrive en fin de vie (ou qu'il est hors d'usage) et à cet effet, vous êtes tenu de le remettre à un point de collecte agréé pour le recyclage des équipements électriques et électroniques usagés. Le tri, l'évacuation et le recyclage séparés de vos équipements usagés permettent de préserver les ressources naturelles et de s'assurer que ces équipements sont recyclés dans le respect de la santé humaine et de l'environnement. Pour plus d'informations sur les lieux de collecte des équipements électroniques usagés, contacter votre mairie ou votre service local de traitement des déchets.

12 History

Date	Version	Comment	Author
30-Oct.-2013	1.0	○ Initial Release.	Jack Kim
25-May-2022	1.1	○ Added certificate related functions.	Jack Kim