

libmbus

Generated by Doxygen 1.7.1

Sun Nov 6 2011 08:25:34

Contents

1	libmbus	1
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	<code>_mbus_address</code> Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Field Documentation	7
4.1.2.1	<code>"@3</code>	7
4.1.2.2	<code>is_primary</code>	7
4.1.2.3	<code>primary</code>	7
4.1.2.4	<code>secondary</code>	8
4.2	<code>_mbus_data_fixed</code> Struct Reference	8
4.2.1	Field Documentation	8
4.2.1.1	<code>cnt1_type</code>	8
4.2.1.2	<code>cnt1_val</code>	8
4.2.1.3	<code>cnt2_type</code>	8
4.2.1.4	<code>cnt2_val</code>	8
4.2.1.5	<code>id_bcd</code>	8
4.2.1.6	<code>status</code>	8
4.2.1.7	<code>tx_cnt</code>	8
4.3	<code>_mbus_data_information_block</code> Struct Reference	8
4.3.1	Field Documentation	9
4.3.1.1	<code>dif</code>	9
4.3.1.2	<code>dife</code>	9

4.3.1.3	ndife	9
4.4	_mbus_data_record Struct Reference	9
4.4.1	Field Documentation	9
4.4.1.1	data	9
4.4.1.2	data_len	9
4.4.1.3	drh	9
4.4.1.4	next	9
4.5	_mbus_data_record_header Struct Reference	9
4.5.1	Field Documentation	10
4.5.1.1	dib	10
4.5.1.2	vib	10
4.6	_mbus_data_secondary_address Struct Reference	10
4.6.1	Field Documentation	10
4.6.1.1	id_bcd	10
4.6.1.2	manufacturer	10
4.6.1.3	medium	10
4.6.1.4	version	10
4.7	_mbus_data_variable Struct Reference	10
4.7.1	Field Documentation	11
4.7.1.1	data	11
4.7.1.2	data_len	11
4.7.1.3	header	11
4.7.1.4	mdh	11
4.7.1.5	mfg_data	11
4.7.1.6	mfg_data_len	11
4.7.1.7	nrecords	11
4.7.1.8	record	11
4.8	_mbus_data_variable_header Struct Reference	11
4.8.1	Field Documentation	12
4.8.1.1	access_no	12
4.8.1.2	id_bcd	12
4.8.1.3	manufacturer	12
4.8.1.4	medium	12
4.8.1.5	signature	12
4.8.1.6	status	12
4.8.1.7	version	12

4.9	<code>_mbus_frame</code> Struct Reference	12
4.9.1	Field Documentation	13
4.9.1.1	<code>address</code>	13
4.9.1.2	<code>checksum</code>	13
4.9.1.3	<code>control</code>	13
4.9.1.4	<code>control_information</code>	13
4.9.1.5	<code>data</code>	13
4.9.1.6	<code>data_size</code>	13
4.9.1.7	<code>length1</code>	13
4.9.1.8	<code>length2</code>	13
4.9.1.9	<code>start1</code>	13
4.9.1.10	<code>start2</code>	13
4.9.1.11	<code>stop</code>	13
4.9.1.12	<code>type</code>	13
4.10	<code>_mbus_frame_data</code> Struct Reference	13
4.10.1	Field Documentation	13
4.10.1.1	<code>data_fix</code>	13
4.10.1.2	<code>data_var</code>	13
4.10.1.3	<code>type</code>	13
4.11	<code>_mbus_handle</code> Struct Reference	14
4.11.1	Detailed Description	14
4.11.2	Field Documentation	14
4.11.2.1	<code>"@1</code>	14
4.11.2.2	<code>is_serial</code>	14
4.11.2.3	<code>m_serial_handle</code>	14
4.11.2.4	<code>m_tcp_handle</code>	14
4.12	<code>_mbus_record</code> Struct Reference	15
4.12.1	Detailed Description	15
4.12.2	Field Documentation	15
4.12.2.1	<code>function_medium</code>	15
4.12.2.2	<code>is_numeric</code>	15
4.12.2.3	<code>quantity</code>	15
4.12.2.4	<code>unit</code>	15
4.12.2.5	<code>value</code>	16
4.13	<code>_mbus_serial_handle</code> Struct Reference	16
4.13.1	Field Documentation	16

4.13.1.1	device	16
4.13.1.2	fd	16
4.13.1.3	t	16
4.14	_mbus_slave_data Struct Reference	16
4.14.1	Field Documentation	16
4.14.1.1	state_acd	16
4.14.1.2	state_fcb	16
4.15	_mbus_string Struct Reference	17
4.15.1	Detailed Description	17
4.15.2	Field Documentation	17
4.15.2.1	size	17
4.15.2.2	value	17
4.16	_mbus_tcp_handle Struct Reference	17
4.16.1	Field Documentation	18
4.16.1.1	host	18
4.16.1.2	port	18
4.16.1.3	sock	18
4.17	_mbus_value Union Reference	18
4.17.1	Detailed Description	18
4.17.2	Field Documentation	18
4.17.2.1	real_val	18
4.17.2.2	str_val	18
4.18	_mbus_value_information_block Struct Reference	18
4.18.1	Field Documentation	19
4.18.1.1	nvife	19
4.18.1.2	vif	19
4.18.1.3	vife	19
4.19	_mbus_variable_vif Struct Reference	19
4.19.1	Field Documentation	19
4.19.1.1	exponent	19
4.19.1.2	quantity	19
4.19.1.3	unit	19
4.19.1.4	vif	19
5	File Documentation	21
5.1	mbus/mbus-protocol-aux.c File Reference	21
5.1.1	Define Documentation	23

5.1.1.1	MBUS_DEBUG	23
5.1.1.2	MBUS_ERROR	23
5.1.2	Typedef Documentation	23
5.1.2.1	mbus_variable_vif	23
5.1.3	Function Documentation	23
5.1.3.1	mbus_connect_serial	23
5.1.3.2	mbus_connect_tcp	23
5.1.3.3	mbus_disconnect	23
5.1.3.4	mbus_fixed_normalize	24
5.1.3.5	mbus_parse_fixed_record	24
5.1.3.6	mbus_parse_variable_record	24
5.1.3.7	mbus_probe_secondary_address	24
5.1.3.8	mbus_read_slave	25
5.1.3.9	mbus_record_free	25
5.1.3.10	mbus_record_new	25
5.1.3.11	mbus_recv_frame	25
5.1.3.12	mbus_scan_2nd_address_range	26
5.1.3.13	mbus_send_frame	26
5.1.3.14	mbus_send_ping_frame	26
5.1.3.15	mbus_send_request_frame	26
5.1.3.16	mbus_send_select_frame	26
5.1.3.17	mbus_variable_value_decode	26
5.1.3.18	mbus_vib_unit_normalize	26
5.1.3.19	mbus_vif_unit_normalize	27
5.1.4	Variable Documentation	27
5.1.4.1	fixed_table	27
5.1.4.2	vif_table	27
5.2	mbus/mbus-protocol-aux.h File Reference	27
5.2.1	Detailed Description	30
5.2.2	Define Documentation	30
5.2.2.1	MBUS_PROBE_COLLISION	30
5.2.2.2	MBUS_PROBE_ERROR	30
5.2.2.3	MBUS_PROBE_NOTHING	30
5.2.2.4	MBUS_PROBE_SINGLE	30
5.2.3	Typedef Documentation	30
5.2.3.1	mbus_address	30

5.2.3.2	<code>mbus_handle</code>	31
5.2.3.3	<code>mbus_record</code>	31
5.2.3.4	<code>mbus_string</code>	31
5.2.3.5	<code>mbus_value</code>	31
5.2.4	Function Documentation	31
5.2.4.1	<code>mbus_connect_serial</code>	31
5.2.4.2	<code>mbus_connect_tcp</code>	31
5.2.4.3	<code>mbus_data_fixed_normalize</code>	32
5.2.4.4	<code>mbus_data_variable_value_decode</code>	32
5.2.4.5	<code>mbus_disconnect</code>	32
5.2.4.6	<code>mbus_parse_fixed_record</code>	33
5.2.4.7	<code>mbus_parse_variable_record</code>	33
5.2.4.8	<code>mbus_probe_secondary_address</code>	33
5.2.4.9	<code>mbus_read_slave</code>	33
5.2.4.10	<code>mbus_record_free</code>	34
5.2.4.11	<code>mbus_record_new</code>	34
5.2.4.12	<code>mbus_rcv_frame</code>	34
5.2.4.13	<code>mbus_scan_2nd_address_range</code>	34
5.2.4.14	<code>mbus_send_data_request_frame</code>	35
5.2.4.15	<code>mbus_send_frame</code>	35
5.2.4.16	<code>mbus_send_select_frame</code>	35
5.2.4.17	<code>mbus_vib_unit_normalize</code>	35
5.2.4.18	<code>mbus_vif_unit_normalize</code>	36
5.3	<code>mbus/mbus-protocol.c</code> File Reference	36
5.3.1	Define Documentation	39
5.3.1.1	<code>NITEMS</code>	39
5.3.2	Function Documentation	39
5.3.2.1	<code>calc_checksum</code>	39
5.3.2.2	<code>calc_length</code>	40
5.3.2.3	<code>mbus_data_bcd_decode</code>	40
5.3.2.4	<code>mbus_data_bcd_encode</code>	40
5.3.2.5	<code>mbus_data_fixed_function</code>	40
5.3.2.6	<code>mbus_data_fixed_medium</code>	40
5.3.2.7	<code>mbus_data_fixed_parse</code>	40
5.3.2.8	<code>mbus_data_fixed_print</code>	40
5.3.2.9	<code>mbus_data_fixed_unit</code>	40

5.3.2.10	mbus_data_fixed_xml	40
5.3.2.11	mbus_data_int_decode	40
5.3.2.12	mbus_data_int_encode	40
5.3.2.13	mbus_data_long_decode	41
5.3.2.14	mbus_data_manufacturer_encode	41
5.3.2.15	mbus_data_record_append	41
5.3.2.16	mbus_data_record_decode	41
5.3.2.17	mbus_data_record_free	41
5.3.2.18	mbus_data_record_function	41
5.3.2.19	mbus_data_record_new	41
5.3.2.20	mbus_data_record_unit	41
5.3.2.21	mbus_data_record_value	41
5.3.2.22	mbus_data_str_decode	41
5.3.2.23	mbus_data_variable_header_print	41
5.3.2.24	mbus_data_variable_header_xml	41
5.3.2.25	mbus_data_variable_medium_lookup	42
5.3.2.26	mbus_data_variable_parse	42
5.3.2.27	mbus_data_variable_print	42
5.3.2.28	mbus_data_variable_xml	42
5.3.2.29	mbus_decode_manufacturer	42
5.3.2.30	mbus_dif_datalength_lookup	42
5.3.2.31	mbus_error_reset	42
5.3.2.32	mbus_error_str	42
5.3.2.33	mbus_error_str_set	42
5.3.2.34	mbus_frame_calc_checksum	42
5.3.2.35	mbus_frame_calc_length	42
5.3.2.36	mbus_frame_data_free	42
5.3.2.37	mbus_frame_data_new	43
5.3.2.38	mbus_frame_data_parse	43
5.3.2.39	mbus_frame_data_print	43
5.3.2.40	mbus_frame_data_xml	43
5.3.2.41	mbus_frame_free	43
5.3.2.42	mbus_frame_get_secondary_address	43
5.3.2.43	mbus_frame_internal_pack	43
5.3.2.44	mbus_frame_new	43
5.3.2.45	mbus_frame_pack	43

5.3.2.46	<code>mbus_frame_print</code>	43
5.3.2.47	<code>mbus_frame_select_secondary_pack</code>	43
5.3.2.48	<code>mbus_frame_type</code>	43
5.3.2.49	<code>mbus_frame_verify</code>	44
5.3.2.50	<code>mbus_parse</code>	44
5.3.2.51	<code>mbus_slave_data_get</code>	44
5.3.2.52	<code>mbus_unit_prefix</code>	44
5.3.2.53	<code>mbus_vib_unit_lookup</code>	44
5.3.2.54	<code>mbus_vif_unit_lookup</code>	44
5.3.3	Variable Documentation	44
5.3.3.1	<code>error_str</code>	44
5.3.3.2	<code>parse_debug</code>	44
5.3.3.3	<code>slave_data</code>	44
5.4	<code>mbus/mbus-protocol.h</code> File Reference	44
5.4.1	Detailed Description	50
5.4.2	Define Documentation	52
5.4.2.1	<code>MBUS_ADDRESS_BROADCAST_NOREPLY</code>	52
5.4.2.2	<code>MBUS_ADDRESS_BROADCAST_REPLY</code>	52
5.4.2.3	<code>MBUS_ADDRESS_NETWORK_LAYER</code>	52
5.4.2.4	<code>MBUS_CONTROL_FIELD_ACD</code>	52
5.4.2.5	<code>MBUS_CONTROL_FIELD_DFC</code>	52
5.4.2.6	<code>MBUS_CONTROL_FIELD_DIRECTION</code>	52
5.4.2.7	<code>MBUS_CONTROL_FIELD_F0</code>	52
5.4.2.8	<code>MBUS_CONTROL_FIELD_F1</code>	52
5.4.2.9	<code>MBUS_CONTROL_FIELD_F2</code>	52
5.4.2.10	<code>MBUS_CONTROL_FIELD_F3</code>	52
5.4.2.11	<code>MBUS_CONTROL_FIELD_FCB</code>	52
5.4.2.12	<code>MBUS_CONTROL_FIELD_FCV</code>	52
5.4.2.13	<code>MBUS_CONTROL_INFO_APPLICATION_RESET</code>	52
5.4.2.14	<code>MBUS_CONTROL_INFO_DATA_SEND</code>	52
5.4.2.15	<code>MBUS_CONTROL_INFO_DATA_SEND_MSB</code>	52
5.4.2.16	<code>MBUS_CONTROL_INFO_EEPROM_READ</code>	52
5.4.2.17	<code>MBUS_CONTROL_INFO_INIT_TEST_CALIB</code>	52
5.4.2.18	<code>MBUS_CONTROL_INFO_REQUEST_RAM_READ</code>	52
5.4.2.19	<code>MBUS_CONTROL_INFO_RESP_FIXED</code>	52
5.4.2.20	<code>MBUS_CONTROL_INFO_RESP_FIXED_MSB</code>	52

5.4.2.21	MBUS_CONTROL_INFO_RESP_VARIABLE	52
5.4.2.22	MBUS_CONTROL_INFO_RESP_VARIABLE_MSB	52
5.4.2.23	MBUS_CONTROL_INFO_SELECT_SLAVE	52
5.4.2.24	MBUS_CONTROL_INFO_SELECT_SLAVE_MSB	52
5.4.2.25	MBUS_CONTROL_INFO_SEND_USER_DATA	52
5.4.2.26	MBUS_CONTROL_INFO_SET_BAUDRATE_1200	52
5.4.2.27	MBUS_CONTROL_INFO_SET_BAUDRATE_19200	52
5.4.2.28	MBUS_CONTROL_INFO_SET_BAUDRATE_2400	52
5.4.2.29	MBUS_CONTROL_INFO_SET_BAUDRATE_300	52
5.4.2.30	MBUS_CONTROL_INFO_SET_BAUDRATE_38400	52
5.4.2.31	MBUS_CONTROL_INFO_SET_BAUDRATE_4800	52
5.4.2.32	MBUS_CONTROL_INFO_SET_BAUDRATE_600	52
5.4.2.33	MBUS_CONTROL_INFO_SET_BAUDRATE_9600	52
5.4.2.34	MBUS_CONTROL_INFO_SW_TEST_START	52
5.4.2.35	MBUS_CONTROL_INFO_SYNC_ACTION	52
5.4.2.36	MBUS_CONTROL_MASK_ACD	52
5.4.2.37	MBUS_CONTROL_MASK_DFC	52
5.4.2.38	MBUS_CONTROL_MASK_DIR	52
5.4.2.39	MBUS_CONTROL_MASK_DIR_M2S	52
5.4.2.40	MBUS_CONTROL_MASK_DIR_S2M	52
5.4.2.41	MBUS_CONTROL_MASK_FCB	52
5.4.2.42	MBUS_CONTROL_MASK_FCV	52
5.4.2.43	MBUS_CONTROL_MASK_REQ_UD1	52
5.4.2.44	MBUS_CONTROL_MASK_REQ_UD2	52
5.4.2.45	MBUS_CONTROL_MASK_RSP_UD	52
5.4.2.46	MBUS_CONTROL_MASK_SND_NKE	52
5.4.2.47	MBUS_CONTROL_MASK_SND_UD	52
5.4.2.48	MBUS_DATA_FIXED_STATUS_DATE_CURRENT	52
5.4.2.49	MBUS_DATA_FIXED_STATUS_DATE_MASK	52
5.4.2.50	MBUS_DATA_FIXED_STATUS_DATE_STORED	52
5.4.2.51	MBUS_DATA_FIXED_STATUS_FORMAT_BCD	52
5.4.2.52	MBUS_DATA_FIXED_STATUS_FORMAT_INT	52
5.4.2.53	MBUS_DATA_FIXED_STATUS_FORMAT_MASK	52
5.4.2.54	MBUS_DATA_RECORD_DIF_MASK_EXTENTION	52
5.4.2.55	MBUS_DATA_RECORD_DIF_MASK_INST	52
5.4.2.56	MBUS_DATA_RECORD_DIF_MASK_MIN	52

5.4.2.57	MBUS_DATA_RECORD_DIF_MASK_STORAGE_NO	52
5.4.2.58	MBUS_DATA_RECORD_DIF_MASK_TYPE_INT32	52
5.4.2.59	MBUS_DATA_TYPE_FIXED	52
5.4.2.60	MBUS_DATA_TYPE_VARIABLE	52
5.4.2.61	MBUS_DIB_DIF_EXTENSION_BIT	52
5.4.2.62	MBUS_DIB_VIF_EXTENSION_BIT	52
5.4.2.63	MBUS_FRAME_ACK_BASE_SIZE	52
5.4.2.64	MBUS_FRAME_ACK_START	52
5.4.2.65	MBUS_FRAME_BASE_SIZE_ACK	52
5.4.2.66	MBUS_FRAME_BASE_SIZE_CONTROL	52
5.4.2.67	MBUS_FRAME_BASE_SIZE_LONG	52
5.4.2.68	MBUS_FRAME_BASE_SIZE_SHORT	52
5.4.2.69	MBUS_FRAME_CONTROL_BASE_SIZE	52
5.4.2.70	MBUS_FRAME_CONTROL_START	52
5.4.2.71	MBUS_FRAME_FIXED_SIZE_ACK	52
5.4.2.72	MBUS_FRAME_FIXED_SIZE_CONTROL	52
5.4.2.73	MBUS_FRAME_FIXED_SIZE_LONG	52
5.4.2.74	MBUS_FRAME_FIXED_SIZE_SHORT	52
5.4.2.75	MBUS_FRAME_LONG_BASE_SIZE	52
5.4.2.76	MBUS_FRAME_LONG_START	52
5.4.2.77	MBUS_FRAME_SHORT_BASE_SIZE	52
5.4.2.78	MBUS_FRAME_SHORT_START	52
5.4.2.79	MBUS_FRAME_STOP	52
5.4.2.80	MBUS_FRAME_TYPE_ACK	52
5.4.2.81	MBUS_FRAME_TYPE_ANY	52
5.4.2.82	MBUS_FRAME_TYPE_CONTROL	52
5.4.2.83	MBUS_FRAME_TYPE_LONG	52
5.4.2.84	MBUS_FRAME_TYPE_SHORT	52
5.4.2.85	MBUS_MAX_PRIMARY_SLAVES	52
5.4.2.86	MBUS_VARIABLE_DATA_MEDIUM_ADC	52
5.4.2.87	MBUS_VARIABLE_DATA_MEDIUM_BUS	52
5.4.2.88	MBUS_VARIABLE_DATA_MEDIUM_COLD_WATER	52
5.4.2.89	MBUS_VARIABLE_DATA_MEDIUM_COMPR_AIR	52
5.4.2.90	MBUS_VARIABLE_DATA_MEDIUM_COOL_IN	52
5.4.2.91	MBUS_VARIABLE_DATA_MEDIUM_COOL_OUT	52
5.4.2.92	MBUS_VARIABLE_DATA_MEDIUM_DUAL_WATER	52

5.4.2.93	MBUS_VARIABLE_DATA_MEDIUM_ELECTRICITY	52
5.4.2.94	MBUS_VARIABLE_DATA_MEDIUM_GAS	52
5.4.2.95	MBUS_VARIABLE_DATA_MEDIUM_HEAT	52
5.4.2.96	MBUS_VARIABLE_DATA_MEDIUM_HEAT_COST	52
5.4.2.97	MBUS_VARIABLE_DATA_MEDIUM_HOT_WATER	52
5.4.2.98	MBUS_VARIABLE_DATA_MEDIUM_OIL	52
5.4.2.99	MBUS_VARIABLE_DATA_MEDIUM_OTHER	52
5.4.2.100	MBUS_VARIABLE_DATA_MEDIUM_PRESSURE	52
5.4.2.101	MBUS_VARIABLE_DATA_MEDIUM_STEAM	52
5.4.2.102	MBUS_VARIABLE_DATA_MEDIUM_WATER	52
5.4.2.103	NITEMS	52
5.4.3	Typedef Documentation	52
5.4.3.1	mbus_data_fixed	52
5.4.3.2	mbus_data_information_block	52
5.4.3.3	mbus_data_record	52
5.4.3.4	mbus_data_record_header	52
5.4.3.5	mbus_data_secondary_address	52
5.4.3.6	mbus_data_variable	52
5.4.3.7	mbus_data_variable_header	52
5.4.3.8	mbus_frame	52
5.4.3.9	mbus_frame_data	52
5.4.3.10	mbus_slave_data	52
5.4.3.11	mbus_value_information_block	52
5.4.4	Function Documentation	52
5.4.4.1	mbus_data_bcd_decode	52
5.4.4.2	mbus_data_bcd_encode	53
5.4.4.3	mbus_data_fixed_function	53
5.4.4.4	mbus_data_fixed_medium	53
5.4.4.5	mbus_data_fixed_parse	53
5.4.4.6	mbus_data_fixed_print	53
5.4.4.7	mbus_data_fixed_unit	53
5.4.4.8	mbus_data_fixed_xml	53
5.4.4.9	mbus_data_int_decode	53
5.4.4.10	mbus_data_int_encode	53
5.4.4.11	mbus_data_long_decode	53
5.4.4.12	mbus_data_manufacturer_encode	53

5.4.4.13	<code>mbus_data_record_append</code>	53
5.4.4.14	<code>mbus_data_record_free</code>	54
5.4.4.15	<code>mbus_data_record_function</code>	54
5.4.4.16	<code>mbus_data_record_new</code>	54
5.4.4.17	<code>mbus_data_str_decode</code>	54
5.4.4.18	<code>mbus_data_variable_header_print</code>	54
5.4.4.19	<code>mbus_data_variable_header_xml</code>	54
5.4.4.20	<code>mbus_data_variable_medium_lookup</code>	54
5.4.4.21	<code>mbus_data_variable_parse</code>	54
5.4.4.22	<code>mbus_data_variable_print</code>	54
5.4.4.23	<code>mbus_data_variable_xml</code>	54
5.4.4.24	<code>mbus_data_xml</code>	54
5.4.4.25	<code>mbus_decode_manufacturer</code>	54
5.4.4.26	<code>mbus_dif_datalength_lookup</code>	55
5.4.4.27	<code>mbus_error_reset</code>	55
5.4.4.28	<code>mbus_error_str</code>	55
5.4.4.29	<code>mbus_error_str_set</code>	55
5.4.4.30	<code>mbus_frame_calc_checksum</code>	55
5.4.4.31	<code>mbus_frame_calc_length</code>	55
5.4.4.32	<code>mbus_frame_data_free</code>	55
5.4.4.33	<code>mbus_frame_data_new</code>	55
5.4.4.34	<code>mbus_frame_data_parse</code>	55
5.4.4.35	<code>mbus_frame_data_print</code>	55
5.4.4.36	<code>mbus_frame_data_xml</code>	55
5.4.4.37	<code>mbus_frame_free</code>	55
5.4.4.38	<code>mbus_frame_get_secondary_address</code>	56
5.4.4.39	<code>mbus_frame_internal_pack</code>	56
5.4.4.40	<code>mbus_frame_new</code>	56
5.4.4.41	<code>mbus_frame_pack</code>	56
5.4.4.42	<code>mbus_frame_print</code>	56
5.4.4.43	<code>mbus_frame_select_secondary_pack</code>	56
5.4.4.44	<code>mbus_frame_type</code>	56
5.4.4.45	<code>mbus_frame_verify</code>	56
5.4.4.46	<code>mbus_parse</code>	56
5.4.4.47	<code>mbus_slave_data_get</code>	56
5.4.4.48	<code>mbus_unit_prefix</code>	56

5.4.4.49	<code>mbus_vib_unit_lookup</code>	56
5.4.4.50	<code>mbus_vif_unit_lookup</code>	57
5.5	<code>mbus/mbus-serial.c</code> File Reference	57
5.5.1	Define Documentation	57
5.5.1.1	<code>PACKET_BUFF_SIZE</code>	57
5.5.2	Function Documentation	57
5.5.2.1	<code>mbus_serial_connect</code>	57
5.5.2.2	<code>mbus_serial_disconnect</code>	58
5.5.2.3	<code>mbus_serial_recv_frame</code>	58
5.5.2.4	<code>mbus_serial_send_frame</code>	58
5.6	<code>mbus/mbus-serial.h</code> File Reference	58
5.6.1	Detailed Description	58
5.6.2	Typedef Documentation	58
5.6.2.1	<code>mbus_serial_handle</code>	58
5.6.3	Function Documentation	58
5.6.3.1	<code>mbus_serial_connect</code>	58
5.6.3.2	<code>mbus_serial_disconnect</code>	59
5.6.3.3	<code>mbus_serial_recv_frame</code>	59
5.6.3.4	<code>mbus_serial_send_frame</code>	59
5.7	<code>mbus/mbus-tcp.c</code> File Reference	59
5.7.1	Define Documentation	59
5.7.1.1	<code>PACKET_BUFF_SIZE</code>	59
5.7.2	Function Documentation	59
5.7.2.1	<code>mbus_tcp_connect</code>	59
5.7.2.2	<code>mbus_tcp_disconnect</code>	60
5.7.2.3	<code>mbus_tcp_recv_frame</code>	60
5.7.2.4	<code>mbus_tcp_send_frame</code>	60
5.8	<code>mbus/mbus-tcp.h</code> File Reference	60
5.8.1	Detailed Description	60
5.8.2	Typedef Documentation	60
5.8.2.1	<code>mbus_tcp_handle</code>	60
5.8.3	Function Documentation	60
5.8.3.1	<code>mbus_tcp_connect</code>	60
5.8.3.2	<code>mbus_tcp_disconnect</code>	61
5.8.3.3	<code>mbus_tcp_recv_frame</code>	61
5.8.3.4	<code>mbus_tcp_send_frame</code>	61

5.9	mbus/mbus.c File Reference	61
5.9.1	Function Documentation	61
5.9.1.1	mbus_init	61
5.10	mbus/mbus.h File Reference	61
5.10.1	Detailed Description	61
5.10.2	Function Documentation	61
5.10.2.1	mbus_init	61

Chapter 1

libmbus

These pages contain automatically generated documentation for the libmbus API. For examples on how to use the libmbus library, see the applications in the bin directory in the source code distribution.

For more information, see <http://www.freescada.com/libmbus>

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_mbus_address (MBus slave address type (primary/secondary address))	7
_mbus_data_fixed	8
_mbus_data_information_block	8
_mbus_data_record	9
_mbus_data_record_header	9
_mbus_data_secondary_address	10
_mbus_data_variable	10
_mbus_data_variable_header	11
_mbus_frame	12
_mbus_frame_data	13
_mbus_handle (Unified MBus handle type encapsulating either Serial or TCP gateway)	14
_mbus_record (Single measured quantity record type)	15
_mbus_serial_handle	16
_mbus_slave_data	16
_mbus_string (_string type)	17
_mbus_tcp_handle	17
_mbus_value (Quantity value type union)	18
_mbus_value_information_block	18
_mbus_variable_vif	19

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

mbus/mbus-protocol-aux.c	21
mbus/mbus-protocol-aux.h (Auxiliary functions to the Freescada libmbus library)	27
mbus/mbus-protocol.c	36
mbus/mbus-protocol.h (Functions and data structures for M-Bus protocol parsing)	44
mbus/mbus-serial.c	57
mbus/mbus-serial.h (Functions and data structures for sending M-Bus data via RS232)	58
mbus/mbus-tcp.c	59
mbus/mbus-tcp.h (Functions and data structures for sending M-Bus data via TCP)	60
mbus/mbus.c	61
mbus/mbus.h (Main include file for the Freescada libmbus library)	61

Chapter 4

Data Structure Documentation

4.1 `_mbus_address` Struct Reference

Mbus slave address type (primary/secodary address).

```
#include <mbus-protocol-aux.h>
```

Data Fields

- char `is_primary`
Address type (non zero for primary).
- union {
 - int `primary`
Primary address (for slaves shall be from 1 to 250).
 - char * `secondary`
Secondary address (shall be 16 digits).

4.1.1 Detailed Description

Mbus slave address type (primary/secodary address).

4.1.2 Field Documentation

4.1.2.1 `union { ... }`

4.1.2.2 `char _mbus_address::is_primary`

Address type (non zero for primary).

4.1.2.3 `int _mbus_address::primary`

Primary address (for slaves shall be from 1 to 250).

4.1.2.4 char* _mbus_address::secondary

Secondary address (shall be 16 digits).

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

- [mbus/mbus-protocol-aux.h](#)

4.2 _mbus_data_fixed Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- u_char [id_bcd](#) [4]
- u_char [tx_cnt](#)
- u_char [status](#)
- u_char [cnt1_type](#)
- u_char [cnt2_type](#)
- u_char [cnt1_val](#) [4]
- u_char [cnt2_val](#) [4]

4.2.1 Field Documentation

4.2.1.1 u_char _mbus_data_fixed::cnt1_type

4.2.1.2 u_char _mbus_data_fixed::cnt1_val[4]

4.2.1.3 u_char _mbus_data_fixed::cnt2_type

4.2.1.4 u_char _mbus_data_fixed::cnt2_val[4]

4.2.1.5 u_char _mbus_data_fixed::id_bcd[4]

4.2.1.6 u_char _mbus_data_fixed::status

4.2.1.7 u_char _mbus_data_fixed::tx_cnt

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

- [mbus/mbus-protocol.h](#)

4.3 _mbus_data_information_block Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- `u_char dif`
- `u_char dife` [10]
- `size_t ndife`

4.3.1 Field Documentation

4.3.1.1 `u_char _mbus_data_information_block::dif`

4.3.1.2 `u_char _mbus_data_information_block::dife`[10]

4.3.1.3 `size_t _mbus_data_information_block::ndife`

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

- [mbus/mbus-protocol.h](#)

4.4 `_mbus_data_record` Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- `mbus_data_record_header drh`
- `u_char data` [234]
- `size_t data_len`
- `void * next`

4.4.1 Field Documentation

4.4.1.1 `u_char _mbus_data_record::data`[234]

4.4.1.2 `size_t _mbus_data_record::data_len`

4.4.1.3 `mbus_data_record_header _mbus_data_record::drh`

4.4.1.4 `void* _mbus_data_record::next`

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

- [mbus/mbus-protocol.h](#)

4.5 `_mbus_data_record_header` Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- [mbus_data_information_block dib](#)
- [mbus_value_information_block vib](#)

4.5.1 Field Documentation

4.5.1.1 `mbus_data_information_block_mbus_data_record_header::dib`

4.5.1.2 `mbus_value_information_block_mbus_data_record_header::vib`

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

- [mbus/mbus-protocol.h](#)

4.6 `_mbus_data_secondary_address` Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- `u_char` [id_bcd](#) [4]
- `u_char` [manufacturer](#) [2]
- `u_char` [version](#)
- `u_char` [medium](#)

4.6.1 Field Documentation

4.6.1.1 `u_char _mbus_data_secondary_address::id_bcd[4]`

4.6.1.2 `u_char _mbus_data_secondary_address::manufacturer[2]`

4.6.1.3 `u_char _mbus_data_secondary_address::medium`

4.6.1.4 `u_char _mbus_data_secondary_address::version`

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

- [mbus/mbus-protocol.h](#)

4.7 `_mbus_data_variable` Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- [mbus_data_variable_header](#) header
- [mbus_data_record](#) * record
- [size_t](#) nrecords
- [u_char](#) * data
- [size_t](#) data_len
- [u_char](#) mdh
- [u_char](#) * mfg_data
- [size_t](#) mfg_data_len

4.7.1 Field Documentation

4.7.1.1 [u_char* _mbus_data_variable::data](#)

4.7.1.2 [size_t _mbus_data_variable::data_len](#)

4.7.1.3 [mbus_data_variable_header _mbus_data_variable::header](#)

4.7.1.4 [u_char _mbus_data_variable::mdh](#)

4.7.1.5 [u_char* _mbus_data_variable::mfg_data](#)

4.7.1.6 [size_t _mbus_data_variable::mfg_data_len](#)

4.7.1.7 [size_t _mbus_data_variable::nrecords](#)

4.7.1.8 [mbus_data_record* _mbus_data_variable::record](#)

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

- [mbus/mbus-protocol.h](#)

4.8 _mbus_data_variable_header Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- [u_char](#) id_bcd [4]
- [u_char](#) manufacturer [2]
- [u_char](#) version
- [u_char](#) medium
- [u_char](#) access_no
- [u_char](#) status
- [u_char](#) signature [2]

4.8.1 Field Documentation

4.8.1.1 `u_char _mbus_data_variable_header::access_no`

4.8.1.2 `u_char _mbus_data_variable_header::id_bcd[4]`

4.8.1.3 `u_char _mbus_data_variable_header::manufacturer[2]`

4.8.1.4 `u_char _mbus_data_variable_header::medium`

4.8.1.5 `u_char _mbus_data_variable_header::signature[2]`

4.8.1.6 `u_char _mbus_data_variable_header::status`

4.8.1.7 `u_char _mbus_data_variable_header::version`

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

- [mbus/mbus-protocol.h](#)

4.9 `_mbus_frame` Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- `u_char` [start1](#)
- `u_char` [length1](#)
- `u_char` [length2](#)
- `u_char` [start2](#)
- `u_char` [control](#)
- `u_char` [address](#)
- `u_char` [control_information](#)
- `u_char` [checksum](#)
- `u_char` [stop](#)
- `u_char` [data](#) [252]
- `size_t` [data_size](#)
- `int` [type](#)

4.9.1 Field Documentation

- 4.9.1.1 `u_char _mbus_frame::address`
- 4.9.1.2 `u_char _mbus_frame::checksum`
- 4.9.1.3 `u_char _mbus_frame::control`
- 4.9.1.4 `u_char _mbus_frame::control_information`
- 4.9.1.5 `u_char _mbus_frame::data[252]`
- 4.9.1.6 `size_t _mbus_frame::data_size`
- 4.9.1.7 `u_char _mbus_frame::length1`
- 4.9.1.8 `u_char _mbus_frame::length2`
- 4.9.1.9 `u_char _mbus_frame::start1`
- 4.9.1.10 `u_char _mbus_frame::start2`
- 4.9.1.11 `u_char _mbus_frame::stop`
- 4.9.1.12 `int _mbus_frame::type`

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

- [mbus/mbus-protocol.h](#)

4.10 `_mbus_frame_data` Struct Reference

```
#include <mbus-protocol.h>
```

Data Fields

- [mbus_data_variable data_var](#)
- [mbus_data_fixed data_fix](#)
- `int type`

4.10.1 Field Documentation

- 4.10.1.1 `mbus_data_fixed _mbus_frame_data::data_fix`
- 4.10.1.2 `mbus_data_variable _mbus_frame_data::data_var`
- 4.10.1.3 `int _mbus_frame_data::type`

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

- [mbus/mbus-protocol.h](#)

4.11 `_mbus_handle` Struct Reference

Unified MBus handle type encapsulating either Serial or TCP gateway.

```
#include <mbus-protocol-aux.h>
```

Data Fields

- char `is_serial`
_handle type (non zero for serial)
 - union {
 - `mbus_tcp_handle * m_tcp_handle`
TCP gateway handle.
 - `mbus_serial_handle * m_serial_handle`
Serial gateway handle.
- ```
};
```

### 4.11.1 Detailed Description

Unified MBus handle type encapsulating either Serial or TCP gateway.

### 4.11.2 Field Documentation

#### 4.11.2.1 union { ... }

#### 4.11.2.2 char `_mbus_handle::is_serial`

`_handle` type (non zero for serial)

#### 4.11.2.3 `mbus_serial_handle* _mbus_handle::m_serial_handle`

Serial gateway handle.

#### 4.11.2.4 `mbus_tcp_handle* _mbus_handle::m_tcp_handle`

TCP gateway handle.

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

- [mbus/mbus-protocol-aux.h](#)

## 4.12 `_mbus_record` Struct Reference

Single measured quantity record type.

```
#include <mbus-protocol-aux.h>
```

### Data Fields

- `mbus_value` `value`  
*Quantity value.*
- `char is_numeric`  
*Quantity value type (nonzero is numeric).*
- `char * unit`  
*Quantity unit (e.g.*
- `char * function_medium`  
*Quantity medium or function (e.g.*
- `char * quantity`  
*Quantity type (e.g.*

### 4.12.1 Detailed Description

Single measured quantity record type.

### 4.12.2 Field Documentation

#### 4.12.2.1 `char* _mbus_record::function_medium`

Quantity medium or function (e.g.

Electricity)

#### 4.12.2.2 `char _mbus_record::is_numeric`

Quantity value type (nonzero is numeric).

#### 4.12.2.3 `char* _mbus_record::quantity`

Quantity type (e.g.

Energy)

#### 4.12.2.4 `char* _mbus_record::unit`

Quantity unit (e.g.

Wh)

#### 4.12.2.5 `mbus_value _mbus_record::value`

Quantity value.

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

- [mbus/mbus-protocol-aux.h](#)

### 4.13 `_mbus_serial_handle` Struct Reference

```
#include <mbus-serial.h>
```

#### Data Fields

- `char * device`
- `int fd`
- `struct termios t`

#### 4.13.1 Field Documentation

##### 4.13.1.1 `char* _mbus_serial_handle::device`

##### 4.13.1.2 `int _mbus_serial_handle::fd`

##### 4.13.1.3 `struct termios _mbus_serial_handle::t`

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

- [mbus/mbus-serial.h](#)

### 4.14 `_mbus_slave_data` Struct Reference

```
#include <mbus-protocol.h>
```

#### Data Fields

- `int state_fcb`
- `int state_acd`

#### 4.14.1 Field Documentation

##### 4.14.1.1 `int _mbus_slave_data::state_acd`

##### 4.14.1.2 `int _mbus_slave_data::state_fcb`

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

- [mbus/mbus-protocol.h](#)

## 4.15 `_mbus_string` Struct Reference

`_string` type

```
#include <mbus-protocol-aux.h>
```

### Data Fields

- `char * value`

*Buffer.*

- `int size`

*\_size*

### 4.15.1 Detailed Description

`_string` type In general support binary strings (octet string, non zero terminated) but so far almost all strings are zero terminated ("texts").

### 4.15.2 Field Documentation

#### 4.15.2.1 `int _mbus_string::size`

*\_size*

#### 4.15.2.2 `char* _mbus_string::value`

Buffer.

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

- [mbus/mbus-protocol-aux.h](#)

## 4.16 `_mbus_tcp_handle` Struct Reference

```
#include <mbus-tcp.h>
```

### Data Fields

- `char * host`
- `int port`
- `int sock`

### 4.16.1 Field Documentation

4.16.1.1 `char* _mbus_tcp_handle::host`

4.16.1.2 `int _mbus_tcp_handle::port`

4.16.1.3 `int _mbus_tcp_handle::sock`

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

- [mbus/mbus-tcp.h](#)

## 4.17 `_mbus_value` Union Reference

Quantity value type union.

```
#include <mbus-protocol-aux.h>
```

### Data Fields

- double `real_val`  
*Numerical.*
- `mbus_string str_val`  
*Text/binary.*

### 4.17.1 Detailed Description

Quantity value type union.

### 4.17.2 Field Documentation

4.17.2.1 `double _mbus_value::real_val`

Numerical.

4.17.2.2 `mbus_string _mbus_value::str_val`

Text/binary.

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

- [mbus/mbus-protocol-aux.h](#)

## 4.18 `_mbus_value_information_block` Struct Reference

```
#include <mbus-protocol.h>
```

## Data Fields

- `u_char vif`
- `u_char vife` [10]
- `size_t nvife`

### 4.18.1 Field Documentation

4.18.1.1 `size_t _mbus_value_information_block::nvife`

4.18.1.2 `u_char _mbus_value_information_block::vif`

4.18.1.3 `u_char _mbus_value_information_block::vife[10]`

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

- `mbus/mbus-protocol.h`

## 4.19 `_mbus_variable_vif` Struct Reference

### Data Fields

- `unsigned vif`
- `double exponent`
- `const char * unit`
- `const char * quantity`

### 4.19.1 Field Documentation

4.19.1.1 `double _mbus_variable_vif::exponent`

4.19.1.2 `const char* _mbus_variable_vif::quantity`

4.19.1.3 `const char* _mbus_variable_vif::unit`

4.19.1.4 `unsigned _mbus_variable_vif::vif`

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

- `mbus/mbus-protocol-aux.c`



# Chapter 5

## File Documentation

### 5.1 mbus/mbus-protocol-aux.c File Reference

```
#include "mbus-protocol-aux.h"
#include <stdio.h>
#include <string.h>
```

#### Data Structures

- struct [\\_mbus\\_variable\\_vif](#)

#### Defines

- #define [MBUS\\_ERROR\(...\)](#) `fprintf(stderr, __VA_ARGS__)`
- #define [MBUS\\_DEBUG\(...\)](#)

#### Typedefs

- typedef struct [\\_mbus\\_variable\\_vif](#) [mbus\\_variable\\_vif](#)

#### Functions

- int [mbus\\_fixed\\_normalize](#) (int medium\_unit, long medium\_value, char \*\*unit\_out, double \*value\_out, char \*\*quantity\_out)
- int [mbus\\_variable\\_value\\_decode](#) ([mbus\\_data\\_record](#) \*record, double \*value\_out\_real, char \*\*value\_out\_str, int \*value\_out\_str\_size)
- int [mbus\\_vif\\_unit\\_normalize](#) (int vif, double value, char \*\*unit\_out, double \*value\_out, char \*\*quantity\_out)  
*Decode units and normalize value using VIF/VIFE (used internally by [mbus\\_vib\\_unit\\_normalize](#)).*
- int [mbus\\_vib\\_unit\\_normalize](#) ([mbus\\_value\\_information\\_block](#) \*vib, double value, char \*\*unit\_out, double \*value\_out, char \*\*quantity\_out)  
*Decode units and normalize value from VIB.*

- `mbus_record * mbus_record_new ()`  
*Allocate new data record.*
- `void mbus_record_free (mbus_record *rec)`  
*Destructor for mbus\_record.*
- `mbus_record * mbus_parse_fixed_record (char status_byte, char medium_unit, u_char *data)`  
*Create/parse single counter from the fixed data structure.*
- `mbus_record * mbus_parse_variable_record (mbus_data_record *data)`  
*Create/parse single counter from the variable data structure record.*
- `mbus_handle * mbus_connect_serial (const char *device)`  
*Connects to serial gateway and initializes MBus handle.*
- `mbus_handle * mbus_connect_tcp (const char *host, int port)`  
*Connects to TCP gateway and initializes MBus handle.*
- `int mbus_disconnect (mbus_handle *handle)`  
*Disconnects the "unified" handle.*
- `int mbus_recv_frame (mbus_handle *handle, mbus_frame *frame)`  
*Receives a frame using "unified" handle.*
- `int mbus_send_frame (mbus_handle *handle, mbus_frame *frame)`  
*Sends frame using "unified" handle.*
- `int mbus_send_select_frame (mbus_handle *handle, const char *secondary_addr_str)`  
*Sends secondary address selection frame using "unified" handle.*
- `int mbus_send_request_frame (mbus_handle *handle, int address)`
- `int mbus_send_ping_frame (mbus_handle *handle, int address)`
- `int mbus_probe_secondary_address (mbus_handle *handle, const char *mask, char *matching_addr)`  
*Probe/address slave by secondary address using "unified" handle.*
- `int mbus_read_slave (mbus_handle *handle, mbus_address *address, mbus_frame *reply)`  
*Read data from given slave using "unified" handle and address types.*
- `int mbus_scan_2nd_address_range (mbus_handle *handle, int pos, char *addr_mask)`  
*Iterate over secondary addresses, send a probe package to all addresses matching the given addresses mask.*

## Variables

- `mbus_variable_vif vif_table []`
- `mbus_variable_vif fixed_table []`

## 5.1.1 Define Documentation

5.1.1.1 `#define MBUS_DEBUG( ... )`

5.1.1.2 `#define MBUS_ERROR( ... ) fprintf( stderr, __VA_ARGS__ )`

## 5.1.2 Typedef Documentation

5.1.2.1 `typedef struct _mbus_variable_vif mbus_variable_vif`

## 5.1.3 Function Documentation

5.1.3.1 `mbus_handle* mbus_connect_serial( const char * device )`

Connects to serial gateway and initializes MBus handle.

### Parameters

*device* Serial device (like /dev/ttyUSB0 or /dev/ttyS0)

### Returns

Initialized "unified" handler when successful, NULL otherwise;

5.1.3.2 `mbus_handle* mbus_connect_tcp( const char * host, int port )`

Connects to TCP gateway and initializes MBus handle.

### Parameters

*host* Gateway host

*port* Gateway port

### Returns

Initialized "unified" handler when successful, NULL otherwise;

5.1.3.3 `int mbus_disconnect( mbus_handle * handle )`

Disconnects the "unified" handle.

### Parameters

*handle* Initialized handle

### Returns

Zero when successful.

**5.1.3.4** `int mbus_fixed_normalize ( int medium_unit, long medium_value, char ** unit_out, double * value_out, char ** quantity_out )`

**5.1.3.5** `mbus_record* mbus_parse_fixed_record ( char statusByte, char medium_unit_byte, u_char * data )`

Create/parse single counter from the fixed data structure.

#### Parameters

*statusByte* status byte

*medium\_unit\_byte* medium/unit byte

*data* pointer to the data counter (4 bytes)

#### Returns

Newly allocated record if succesful, NULL otherwise. Later on need to use [mbus\\_record\\_free](#)

**5.1.3.6** `mbus_record* mbus_parse_variable_record ( mbus_data_record * record )`

Create/parse single counter from the variable data structure record.

#### Parameters

*data* record data to be parsed

#### Returns

Newly allocated record if succesful, NULL otherwise. Later on need to use [mbus\\_record\\_free](#)

< raw value

< normalized value

**5.1.3.7** `int mbus_probe_secondary_address ( mbus_handle * handle, const char * mask, char * matching_addr )`

Probe/address slave by secondary address using "unified" handle.

#### Parameters

*handle* Initialized handle

*mask* Address/mask to probe

*matching\_addr* Matched address (the buffer has to be at least 16 bytes)

#### Returns

See MBUS\_PROBE\_\* constants

**5.1.3.8** `int mbus_read_slave ( mbus_handle * handle, mbus_address * address, mbus_frame * reply )`

Read data from given slave using "unified" handle and address types.

#### Parameters

*handle* Initialized handle  
*address* Address of the slave  
*reply* Reply from the slave

#### Returns

Zero when successful.

**5.1.3.9** `void mbus_record_free ( mbus_record * rec )`

Destructor for mbus\_record.

#### Parameters

*rec* record to be freed

**5.1.3.10** `mbus_record* mbus_record_new ( )`

Allocate new data record.

Use [mbus\\_record\\_free](#) when finished.

#### Returns

pointer to the new record, NULL when failed

**5.1.3.11** `int mbus_recv_frame ( mbus_handle * handle, mbus_frame * frame )`

Receives a frame using "unified" handle.

#### Parameters

*handle* Initialized handle  
*frame* Received frame

#### Returns

Zero when successful.

**5.1.3.12** `int mbus_scan_2nd_address_range ( mbus_handle * handle, int pos, char * addr_mask )`

Iterate over secondary addresses, send a probe package to all addresses matching the given addresses mask.

**Parameters**

*pos* current address  
*addr\_mask* address mask to

**Returns**

zero when OK

**5.1.3.13** `int mbus_send_frame ( mbus_handle * handle, mbus_frame * frame )`

Sends frame using "unified" handle.

**Parameters**

*handle* Initialized handle  
*frame* Frame to send

**Returns**

Zero when successful.

**5.1.3.14** `int mbus_send_ping_frame ( mbus_handle * handle, int address )`

**5.1.3.15** `int mbus_send_request_frame ( mbus_handle * handle, int address )`

**5.1.3.16** `int mbus_send_select_frame ( mbus_handle * handle, const char * secondary_addr_str )`

Sends secondary address selection frame using "unified" handle.

**Parameters**

*handle* Initialized handle  
*secondary\_addr\_str* Secondary address

**Returns**

Zero when successful.

**5.1.3.17** `int mbus_variable_value_decode ( mbus_data_record * record, double * value_out_real, char ** value_out_str, int * value_out_str_size )`

**5.1.3.18** `int mbus_vib_unit_normalize ( mbus_value_information_block * vib, double value, char ** unit_out, double * value_out, char ** quantity_out )`

Decode units and normalize value from VIB.

**Parameters**

*vib* mbus value information block of the variable record  
*value* already parsed "raw" numerical value  
*unit\_out* parsed unit, when done use "free"  
*value\_out* normalized value  
*quantity\_out* parsed quantity, when done use "free"

**Returns**

zero when OK

### 5.1.3.19 int mbus\_vif\_unit\_normalize ( int vif, double value, char \*\* unit\_out, double \* value\_out, char \*\* quantity\_out )

Decode units and normalize value using VIF/VIFE (used internally by mbus\_vib\_unit\_normalize).

**Parameters**

*vif* VIF (including standard extensions)  
*value* already parsed "raw" numerical value  
*unit\_out* parsed unit, when done use "free"  
*value\_out* normalized value  
*quantity\_out* parsed quantity, when done use "free"

**Returns**

zero when OK

**5.1.4 Variable Documentation****5.1.4.1 mbus\_variable\_vif fixed\_table[ ]****5.1.4.2 mbus\_variable\_vif vif\_table[ ]****5.2 mbus/mbus-protocol-aux.h File Reference**

Auxiliary functions to the Freescada libmbus library.

```
#include <mbus/mbus.h>
#include <mbus/mbus-protocol.h>
#include <mbus/mbus-serial.h>
#include <mbus/mbus-tcp.h>
```

**Data Structures**

- struct [\\_mbus\\_handle](#)  
*Unified MBus handle type encapsulating either Serial or TCP gateway.*

- struct `_mbus_address`  
*MBus slave address type (primary/secodary address).*
- struct `_mbus_string`  
*\_string type*
- union `_mbus_value`  
*Quantity value type union.*
- struct `_mbus_record`  
*Single measured quantity record type.*

## Defines

- #define `MBUS_PROBE_NOTHING` 0
- #define `MBUS_PROBE_SINGLE` 1
- #define `MBUS_PROBE_COLLISION` 2
- #define `MBUS_PROBE_ERROR` -1

## Typedefs

- typedef struct `_mbus_handle` `mbus_handle`  
*Unified MBus handle type encapsulating either Serial or TCP gateway.*
- typedef struct `_mbus_address` `mbus_address`  
*MBus slave address type (primary/secodary address).*
- typedef struct `_mbus_string` `mbus_string`  
*\_string type*
- typedef union `_mbus_value` `mbus_value`  
*Quantity value type union.*
- typedef struct `_mbus_record` `mbus_record`  
*Single measured quantity record type.*

## Functions

- `mbus_handle * mbus_connect_serial` (const char \*device)  
*Connects to serial gateway and initializes MBus handle.*
- `mbus_handle * mbus_connect_tcp` (const char \*host, int port)  
*Connects to TCP gateway and initializes MBus handle.*
- int `mbus_disconnect` (`mbus_handle` \*handle)  
*Disconnects the "unified" handle.*

- int `mbus_recv_frame` (`mbus_handle` \*handle, `mbus_frame` \*frame)  
*Receives a frame using "unified" handle.*
- int `mbus_send_frame` (`mbus_handle` \*handle, `mbus_frame` \*frame)  
*Sends frame using "unified" handle.*
- int `mbus_send_select_frame` (`mbus_handle` \*handle, const char \*secondary\_addr\_str)  
*Sends secondary address selection frame using "unified" handle.*
- int `mbus_send_data_request_frame` (`mbus_handle` \*handle, int address)  
*Sends request frame to given slave using "unified" handle.*
- int `mbus_probe_secondary_address` (`mbus_handle` \*handle, const char \*mask, char \*matching\_addr)  
*Probe/address slave by secondary address using "unified" handle.*
- int `mbus_read_slave` (`mbus_handle` \*handle, `mbus_address` \*address, `mbus_frame` \*reply)  
*Read data from given slave using "unified" handle and address types.*
- `mbus_record` \* `mbus_record_new` ()  
*Allocate new data record.*
- void `mbus_record_free` (`mbus_record` \*rec)  
*Destructor for mbus\_record.*
- `mbus_record` \* `mbus_parse_fixed_record` (char statusByte, char medium\_unit\_byte, u\_char \*data)  
*Create/parse single counter from the fixed data structure.*
- `mbus_record` \* `mbus_parse_variable_record` (`mbus_data_record` \*record)  
*Create/parse single counter from the variable data structure record.*
- int `mbus_data_fixed_normalize` (int medium\_unit\_byte, long medium\_value, char \*\*unit\_out, double \*value\_out, char \*\*quantity\_out)  
*Get normalized counter value for a fixed counter.*
- int `mbus_data_variable_value_decode` (`mbus_record` \*record, double \*value\_out\_real, char \*\*value\_out\_str, int \*value\_out\_str\_size)  
*Decode value of a variable data structure.*
- int `mbus_vif_unit_normalize` (int vif, double value, char \*\*unit\_out, double \*value\_out, char \*\*quantity\_out)  
*Decode units and normalize value using VIF/VIFE (used internally by mbus\_vib\_unit\_normalize).*
- int `mbus_vib_unit_normalize` (`mbus_value_information_block` \*vib, double value, char \*\*unit\_out, double \*value\_out, char \*\*quantity\_out)  
*Decode units and normalize value from VIB.*
- int `mbus_scan_2nd_address_range` (`mbus_handle` \*handle, int pos, char \*addr\_mask)  
*Iterate over secondary addresses, send a probe package to all addresses matching the given addresses mask.*

## 5.2.1 Detailed Description

Auxiliary functions to the Freescada libmbus library. The idea is to simplify the basic task of querying MBus slaves and the data processing. Typical use might be (in oversimplified "pseudocode"):

```
* mbus_handle = mbus_connect_serial(device);
* or
* mbus_handle = mbus_connect_tcp(host, port);
*
* ...
*
* mbus_read_slave(mbus_handle, addresses, &reply);
* mbus_frame_data_parse(reply, &frameData);
* // check the header / record type (fixed/variable)
*
* // for fixed use mbus_data_fixed_medium and 2x mbus_parse_fixed_record
* mbus_data_fixed_medium
* mbusRecord = mbus_parse_fixed_record() // first record
* // process mbusRecord
* mbusRecord = mbus_parse_fixed_record() // second record
* // process mbusRecord
*
* // for variable use mbus_parse_variable_record
* for each record
* mbusRecord = mbus_parse_variable_record(record)
* // process mbusRecord
*
* ...
*
* mbus_disconnect(mbus_handle);
*
```

Note that the quantity values are partially "normalized". For example energy is in Wh even when originally used "decimal" prefixes like MWh. This seems to be sensible as it make easier any processing of a dataset with huge fluctuation (no need to lookup/convert the prefixes). Also the potential conversion to desired units is pretty easy.

On the other hand we acknowledge that use of certain units are expected so we do not convert all units to Si (i.e. no conversion from J to Wh or Bar to Pa.)

## 5.2.2 Define Documentation

**5.2.2.1 #define MBUS\_PROBE\_COLLISION 2**

**5.2.2.2 #define MBUS\_PROBE\_ERROR -1**

**5.2.2.3 #define MBUS\_PROBE\_NOTHING 0**

**5.2.2.4 #define MBUS\_PROBE\_SINGLE 1**

## 5.2.3 Typedef Documentation

**5.2.3.1 typedef struct \_mbus\_address mbus\_address**

MBus slave address type (primary/secodary address).

### 5.2.3.2 typedef struct `_mbus_handle` `mbus_handle`

Unified MBus handle type encapsulating either Serial or TCP gateway.

### 5.2.3.3 typedef struct `_mbus_record` `mbus_record`

Single measured quantity record type.

### 5.2.3.4 typedef struct `_mbus_string` `mbus_string`

`_string` type

In general support binary strings (octet string, non zero terminated) but so far almost all strings are zero terminated ("texts").

### 5.2.3.5 typedef union `_mbus_value` `mbus_value`

Quantity value type union.

## 5.2.4 Function Documentation

### 5.2.4.1 `mbus_handle*` `mbus_connect_serial` ( `const char *` *device* )

Connects to serial gateway and initializes MBus handle.

#### Parameters

*device* Serial device (like `/dev/ttyUSB0` or `/dev/ttyS0`)

#### Returns

Initialized "unified" handler when successful, NULL otherwise;

### 5.2.4.2 `mbus_handle*` `mbus_connect_tcp` ( `const char *` *host*, `int` *port* )

Connects to TCP gateway and initializes MBus handle.

#### Parameters

*host* Gateway host

*port* Gateway port

#### Returns

Initialized "unified" handler when successful, NULL otherwise;

#### 5.2.4.3 `int mbus_data_fixed_normalize ( int medium_unit_byte, long medium_value, char **unit_out, double * value_out, char ** quantity_out )`

Get normalized counter value for a fixed counter.

Get "normalized" value and unit of the counter

##### Parameters

*medium\_unit\_byte* medium/unit byte of the fixed counter

*medium\_value* raw counter value

*unit\_out* units of the counter - use free when done

*value\_out* resulting counter value

*quantity\_out* parsed quantity, when done use "free"

##### Returns

zero when OK

#### 5.2.4.4 `int mbus_data_variable_value_decode ( mbus_record * record, double * value_out_real, char ** value_out_str, int * value_out_str_size )`

Decode value of a variable data structure.

##### Parameters

*record* record to be decoded

*value\_out\_real* numerical counter value output (when numerical)

*value\_out\_str* string counter value output (when string, NULL otherwise), when finished use "free \*value\_out\_str"

*value\_out\_str\_size* string counter value size

##### Returns

zero when OK

#### 5.2.4.5 `int mbus_disconnect ( mbus_handle * handle )`

Disconnects the "unified" handle.

##### Parameters

*handle* Initialized handle

##### Returns

Zero when successful.

#### 5.2.4.6 `mbus_record* mbus_parse_fixed_record ( char statusByte, char medium_unit_byte, u_char * data )`

Create/parse single counter from the fixed data structure.

##### Parameters

*statusByte* status byte  
*medium\_unit\_byte* medium/unit byte  
*data* pointer to the data counter (4 bytes)

##### Returns

Newly allocated record if succesful, NULL otherwise. Later on need to use [mbus\\_record\\_free](#)

#### 5.2.4.7 `mbus_record* mbus_parse_variable_record ( mbus_data_record * record )`

Create/parse single counter from the variable data structure record.

##### Parameters

*data* record data to be parsed

##### Returns

Newly allocated record if succesful, NULL otherwise. Later on need to use [mbus\\_record\\_free](#)

< raw value

< normalized value

#### 5.2.4.8 `int mbus_probe_secondary_address ( mbus_handle * handle, const char * mask, char * matching_addr )`

Probe/address slave by secondary address using "unified" handle.

##### Parameters

*handle* Initialized handle  
*mask* Address/mask to probe  
*matching\_addr* Matched address (the buffer has to be at least 16 bytes)

##### Returns

See MBUS\_PROBE\_\* constants

#### 5.2.4.9 `int mbus_read_slave ( mbus_handle * handle, mbus_address * address, mbus_frame * reply )`

Read data from given slave using "unified" handle and address types.

**Parameters**

*handle* Initialized handle  
*address* Address of the slave  
*reply* Reply from the slave

**Returns**

Zero when successful.

**5.2.4.10 void mbus\_record\_free ( mbus\_record \* rec )**

Destructor for mbus\_record.

**Parameters**

*rec* record to be freed

**5.2.4.11 mbus\_record\* mbus\_record\_new ( )**

Allocate new data record.

Use [mbus\\_record\\_free](#) when finished.

**Returns**

pointer to the new record, NULL when failed

**5.2.4.12 int mbus\_recv\_frame ( mbus\_handle \* handle, mbus\_frame \* frame )**

Receives a frame using "unified" handle.

**Parameters**

*handle* Initialized handle  
*frame* Received frame

**Returns**

Zero when successful.

**5.2.4.13 int mbus\_scan\_2nd\_address\_range ( mbus\_handle \* handle, int pos, char \* addr\_mask )**

Iterate over secondary addresses, send a probe package to all addresses matching the given addresses mask.

**Parameters**

*pos* current address  
*addr\_mask* address mask to

**Returns**

zero when OK

**5.2.4.14** `int mbus_send_data_request_frame ( mbus_handle * handle, int address )`

Sends request frame to given slave using "unified" handle.

**Parameters**

*handle* Initialized handle

*address* Address (0-255)

**Returns**

Zero when successful.

**5.2.4.15** `int mbus_send_frame ( mbus_handle * handle, mbus_frame * frame )`

Sends frame using "unified" handle.

**Parameters**

*handle* Initialized handle

*frame* Frame to send

**Returns**

Zero when successful.

**5.2.4.16** `int mbus_send_select_frame ( mbus_handle * handle, const char * secondary_addr_str )`

Sends secondary address selection frame using "unified" handle.

**Parameters**

*handle* Initialized handle

*secondary\_addr\_str* Secondary address

**Returns**

Zero when successful.

**5.2.4.17** `int mbus_vib_unit_normalize ( mbus_value_information_block * vib, double value, char ** unit_out, double * value_out, char ** quantity_out )`

Decode units and normalize value from VIB.

**Parameters**

*vib* mbus value information block of the variable record

*value* already parsed "raw" numerical value

*unit\_out* parsed unit, when done use "free"

*value\_out* normalized value

*quantity\_out* parsed quantity, when done use "free"

**Returns**

zero when OK

#### 5.2.4.18 `int mbus_vif_unit_normalize ( int vif, double value, char ** unit_out, double * value_out, char ** quantity_out )`

Decode units and normalize value using VIF/VIFE (used internally by `mbus_vib_unit_normalize`).

##### Parameters

- vif* VIF (including standard extensions)
- value* already parsed "raw" numerical value
- unit\_out* parsed unit, when done use "free"
- value\_out* normalized value
- quantity\_out* parsed quantity, when done use "free"

##### Returns

zero when OK

## 5.3 mbus/mbus-protocol.c File Reference

```
#include <assert.h>
#include <stdio.h>
#include <string.h>
#include <mbus/mbus-protocol.h>
```

### Defines

- #define `NITEMS(x)` (`sizeof(x)/sizeof(x[0])`)

### Functions

- `char * mbus_error_str ()`  
*Return a string that contains an the latest error message.*
- `void mbus_error_str_set (char *message)`
- `void mbus_error_reset ()`
- `mbus_slave_data * mbus_slave_data_get (size_t i)`  
*Return a pointer to the slave\_data register.*
- `mbus_frame * mbus_frame_new (int frame_type)`  
*Allocate an M-bus frame data structure and initialize it according to which frame type is requested.*
- `int mbus_frame_free (mbus_frame *frame)`  
*Free the memory resources allocated for the M-Bus frame data structure.*
- `u_char calc_checksum (mbus_frame *frame)`  
*Caclulate the checksum of the M-Bus frame. Internal.*

- int `mbus_frame_calc_checksum` (`mbus_frame *frame`)  
*Calculate the checksum of the M-Bus frame.*
- u\_char `calc_length` (`const mbus_frame *frame`)  
*Calculate the values of the lengths fields in the M-Bus frame.*
- int `mbus_frame_calc_length` (`mbus_frame *frame`)  
*Calculate the values of the lengths fields in the M-Bus frame.*
- int `mbus_frame_type` (`mbus_frame *frame`)  
*Return the M-Bus frame type.*
- int `mbus_frame_verify` (`mbus_frame *frame`)  
*Verify that parsed frame is a valid M-bus frame.*
- int `mbus_data_bcd_encode` (`u_char *bcd_data`, `size_t bcd_data_size`, `int value`)  
*Encode BCD data.*
- long `mbus_data_bcd_decode` (`u_char *bcd_data`, `size_t bcd_data_size`)  
*Decode BCD data.*
- int `mbus_data_int_decode` (`u_char *int_data`, `size_t int_data_size`)  
*Decode INTEGER data.*
- long `mbus_data_long_decode` (`u_char *int_data`, `size_t int_data_size`)
- int `mbus_data_int_encode` (`u_char *int_data`, `size_t int_data_size`, `int value`)  
*Encode INTEGER data (into 'int\_data\_size' bytes).*
- void `mbus_data_str_decode` (`u_char *dst`, `const u_char *src`, `size_t len`)  
*Decode string data.*
- int `mbus_data_manufacturer_encode` (`u_char *m_data`, `u_char *m_code`)  
*Generate manufacturer code from 2-byte encoded data.*
- const char \* `mbus_decode_manufacturer` (`u_char byte1`, `u_char byte2`)  
*Generate manufacturer code from 2-byte encoded data.*
- const char \* `mbus_data_fixed_medium` (`mbus_data_fixed *data`)  
*For fixed-length frames, get a string describing the medium.*
- const char \* `mbus_data_fixed_unit` (`int medium_unit_byte`)  
*For fixed-length frames, get a string describing the unit of the data.*
- const char \* `mbus_data_variable_medium_lookup` (`u_char medium`)  
*For variable-length frames, returns a string describing the medium.*
- const char \* `mbus_unit_prefix` (`int exp`)  
*Lookup the unit description from a VIF field in a data record.*
- u\_char `mbus_dif_datalength_lookup` (`u_char dif`)

*Look up the data length from a VIF field in the data record.*

- `const char * mbus_vif_unit_lookup (u_char vif)`  
*Look up the unit from a VIF field in the data record.*
- `const char * mbus_vib_unit_lookup (mbus_value_information_block *vib)`  
*Lookup the unit from the VIB (VIF or VIFE).*
- `const char * mbus_data_record_decode (mbus_data_record *record)`  
*Return a string containing the data.*
- `const char * mbus_data_record_unit (mbus_data_record *record)`  
*Return the unit description for a variable-length data record.*
- `const char * mbus_data_record_value (mbus_data_record *record)`  
*Return the value for a variable-length data record.*
- `const char * mbus_data_record_function (mbus_data_record *record)`  
*Return a string containing the function description.*
- `const char * mbus_data_fixed_function (int status)`  
*For fixed-length frames, return a string describing the type of value (stored or actual).*
- `int mbus_parse (mbus_frame *frame, u_char *data, size_t data_size)`  
*PARSE M-BUS frame data structures from binary data.*
- `int mbus_data_fixed_parse (mbus_frame *frame, mbus_data_fixed *data)`  
*Parse the fixed-length data of a M-Bus frame.*
- `int mbus_data_variable_parse (mbus_frame *frame, mbus_data_variable *data)`  
*Parse the variable-length data of a M-Bus frame.*
- `int mbus_frame_data_parse (mbus_frame *frame, mbus_frame_data *data)`  
*Check the stype of the frame data (fixed or variable) and dispatch to the corresponding parser function.*
- `int mbus_frame_pack (mbus_frame *frame, u_char *data, size_t data_size)`  
*Pack the M-bus frame into a binary string representation that can be sent on the bus.*
- `int mbus_frame_internal_pack (mbus_frame *frame, mbus_frame_data *frame_data)`  
*pack the data structures into frame->data*
- `int mbus_frame_print (mbus_frame *frame)`  
*Dump frame in HEX on standard output.*
- `int mbus_frame_data_print (mbus_frame_data *data)`  
*Print the data part of a frame.*
- `int mbus_data_variable_header_print (mbus_data_variable_header *header)`  
*Print M-bus frame info to stdout.*

- int `mbus_data_variable_print` (`mbus_data_variable` \*data)
- int `mbus_data_fixed_print` (`mbus_data_fixed` \*data)
- char \* `mbus_data_variable_header_xml` (`mbus_data_variable_header` \*header)  
*Generate XML for the variable-length data header.*
- char \* `mbus_data_variable_xml` (`mbus_data_variable` \*data)  
*Generate XML for variable-length data.*
- char \* `mbus_data_fixed_xml` (`mbus_data_fixed` \*data)  
*Generate XML representation of fixed-length frame.*
- char \* `mbus_frame_data_xml` (`mbus_frame_data` \*data)  
*Return a string containing an XML representation of the M-BUS frame.*
- `mbus_frame_data` \* `mbus_frame_data_new` ()  
*Allocate and initialize a new frame data structure.*
- void `mbus_frame_data_free` (`mbus_frame_data` \*data)  
*Free up data associated with a frame data structure.*
- `mbus_data_record` \* `mbus_data_record_new` ()  
*Allocate and initialize a new variable data record.*
- void `mbus_data_record_free` (`mbus_data_record` \*record)  
*free up memory associated with a data record and all the subsequent records in its list (apply recursively)*
- void `mbus_data_record_append` (`mbus_data_variable` \*data, `mbus_data_record` \*record)  
*Return a string containing an XML representation of the M-BUS frame.*
- char \* `mbus_frame_get_secondary_address` (`mbus_frame` \*frame)
- int `mbus_frame_select_secondary_pack` (`mbus_frame` \*frame, char \*address)

## Variables

- static int `parse_debug` = 0
- static char `error_str` [512]
- static `mbus_slave_data` `slave_data` [MBUS\_MAX\_PRIMARY\_SLAVES]

### 5.3.1 Define Documentation

- 5.3.1.1 `#define` NITEMS( *x* ) (sizeof(x)/sizeof(x[0]))

### 5.3.2 Function Documentation

- 5.3.2.1 `u_char` `calc_checksum` ( `mbus_frame` \* *frame* )

Caclulate the checksum of the M-Bus frame. Internal.

**5.3.2.2** `u_char calc_length ( const mbus_frame * frame )`

Calculate the values of the lengths fields in the M-Bus frame.

Internal.

**5.3.2.3** `long mbus_data_bcd_decode ( u_char * bcd_data, size_t bcd_data_size )`

Decode BCD data.

**5.3.2.4** `int mbus_data_bcd_encode ( u_char * bcd_data, size_t bcd_data_size, int value )`

Encode BCD data.

**5.3.2.5** `const char* mbus_data_fixed_function ( int status )`

For fixed-length frames, return a string describing the type of value (stored or actual).

**5.3.2.6** `const char* mbus_data_fixed_medium ( mbus_data_fixed * data )`

For fixed-length frames, get a string describing the medium.

**5.3.2.7** `int mbus_data_fixed_parse ( mbus_frame * frame, mbus_data_fixed * data )`

Parse the fixed-length data of a M-Bus frame.

**5.3.2.8** `int mbus_data_fixed_print ( mbus_data_fixed * data )`**5.3.2.9** `const char* mbus_data_fixed_unit ( int medium_unit_byte )`

For fixed-length frames, get a string describing the unit of the data.

**5.3.2.10** `char* mbus_data_fixed_xml ( mbus_data_fixed * data )`

Generate XML representation of fixed-length frame.

**5.3.2.11** `int mbus_data_int_decode ( u_char * int_data, size_t int_data_size )`

Decode INTEGER data.

**5.3.2.12** `int mbus_data_int_encode ( u_char * int_data, size_t int_data_size, int value )`

Encode INTEGER data (into 'int\_data\_size' bytes).

**5.3.2.13** `long mbus_data_long_decode ( u_char * int_data, size_t int_data_size )`

**5.3.2.14** `int mbus_data_manufacturer_encode ( u_char * m_data, u_char * m_code )`

Generate manufacturer code from 2-byte encoded data.

**5.3.2.15** `void mbus_data_record_append ( mbus_data_variable * data, mbus_data_record * record )`

Return a string containing an XML representation of the M-BUS frame.

**5.3.2.16** `const char* mbus_data_record_decode ( mbus_data_record * record )`

Return a string containing the data.

**5.3.2.17** `void mbus_data_record_free ( mbus_data_record * record )`

free up memory associated with a data record and all the subsequent records in its list (apply recursively)

**5.3.2.18** `const char* mbus_data_record_function ( mbus_data_record * record )`

Return a string containing the function description.

**5.3.2.19** `mbus_data_record* mbus_data_record_new ( )`

Allocate and initialize a new variable data record.

**5.3.2.20** `const char* mbus_data_record_unit ( mbus_data_record * record )`

Return the unit description for a variable-length data record.

**5.3.2.21** `const char* mbus_data_record_value ( mbus_data_record * record )`

Return the value for a variable-length data record.

**5.3.2.22** `void mbus_data_str_decode ( u_char * dst, const u_char * src, size_t len )`

Decode string data.

**5.3.2.23** `int mbus_data_variable_header_print ( mbus_data_variable_header * header )`

Print M-bus frame info to stdout.

**5.3.2.24** `char* mbus_data_variable_header_xml ( mbus_data_variable_header * header )`

Generate XML for the variable-length data header.

**5.3.2.25** `const char* mbus_data_variable_medium_lookup ( u_char medium )`

For variable-length frames, returns a string describing the medium.

**5.3.2.26** `int mbus_data_variable_parse ( mbus_frame * frame, mbus_data_variable * data )`

Parse the variable-length data of a M-Bus frame.

**5.3.2.27** `int mbus_data_variable_print ( mbus_data_variable * data )`**5.3.2.28** `char* mbus_data_variable_xml ( mbus_data_variable * data )`

Generate XML for variable-length data.

**5.3.2.29** `const char* mbus_decode_manufacturer ( u_char byte1, u_char byte2 )`

Generate manufacturer code from 2-byte encoded data.

**5.3.2.30** `u_char mbus_dif_datalength_lookup ( u_char dif )`

Look up the data length from a VIF field in the data record.

See the table on page 41 the M-BUS specification.

**5.3.2.31** `void mbus_error_reset ( )`**5.3.2.32** `char* mbus_error_str ( )`

Return a string that contains an the latest error message.

**5.3.2.33** `void mbus_error_str_set ( char * message )`**5.3.2.34** `int mbus_frame_calc_checksum ( mbus_frame * frame )`

Calculate the checksum of the M-Bus frame.

The checksum algorithm is the arithmetic sum of the frame content, without using carry. Which content that is included in the checksum calculation depends on the frame type.

**5.3.2.35** `int mbus_frame_calc_length ( mbus_frame * frame )`

Calculate the values of the lengths fields in the M-Bus frame.

**5.3.2.36** `void mbus_frame_data_free ( mbus_frame_data * data )`

Free up data associated with a frame data structure.

**5.3.2.37** `mbus_frame_data* mbus_frame_data_new ( )`

Allocate and initialize a new frame data structure.

**5.3.2.38** `int mbus_frame_data_parse ( mbus_frame * frame, mbus_frame_data * data )`

Check the stype of the frame data (fixed or variable) and dispatch to the corresponding parser function.

**5.3.2.39** `int mbus_frame_data_print ( mbus_frame_data * data )`

Print the data part of a frame.

**5.3.2.40** `char* mbus_frame_data_xml ( mbus_frame_data * data )`

Return a string containing an XML representation of the M-BUS frame.

**5.3.2.41** `int mbus_frame_free ( mbus_frame * frame )`

Free the memory resources allocated for the M-Bus frame data structure.

**5.3.2.42** `char* mbus_frame_get_secondary_address ( mbus_frame * frame )`**5.3.2.43** `int mbus_frame_internal_pack ( mbus_frame * frame, mbus_frame_data * frame_data )`

pack the data structures into frame->data

**5.3.2.44** `mbus_frame* mbus_frame_new ( int frame_type )`

Allocate an M-bus frame data structure and initialize it according to which frame type is requested.

**5.3.2.45** `int mbus_frame_pack ( mbus_frame * frame, u_char * data, size_t data_size )`

Pack the M-bus frame into a binary string representation that can be sent on the bus.

The binary packet format is different for the different types of M-bus frames.

**5.3.2.46** `int mbus_frame_print ( mbus_frame * frame )`

Dump frame in HEX on standard output.

**5.3.2.47** `int mbus_frame_select_secondary_pack ( mbus_frame * frame, char * address )`**5.3.2.48** `int mbus_frame_type ( mbus_frame * frame )`

Return the M-Bus frame type.

**5.3.2.49** `int mbus_frame_verify ( mbus_frame * frame )`

Verify that parsed frame is a valid M-bus frame.

**5.3.2.50** `int mbus_parse ( mbus_frame * frame, u_char * data, size_t data_size )`

PARSE M-BUS frame data structures from binary data.

**5.3.2.51** `mbus_slave_data* mbus_slave_data_get ( size_t i )`

Return a pointer to the slave\_data register.

This register can be used for storing current slave status.

**5.3.2.52** `const char* mbus_unit_prefix ( int exp )`

Lookup the unit description from a VIF field in a data record.

**5.3.2.53** `const char* mbus_vib_unit_lookup ( mbus_value_information_block * vib )`

Lookup the unit from the VIB (VIF or VIFE).

**5.3.2.54** `const char* mbus_vif_unit_lookup ( u_char vif )`

Look up the unit from a VIF field in the data record.

See section 8.4.3 Codes for Value Information Field (VIF) in the M-BUS spec

**5.3.3 Variable Documentation**

**5.3.3.1** `char error_str[512] [static]`

**5.3.3.2** `int parse_debug = 0 [static]`

**5.3.3.3** `mbus_slave_data slave_data[MBUS_MAX_PRIMARY_SLAVES] [static]`

**5.4 mbus/mbus-protocol.h File Reference**

Functions and data structures for M-Bus protocol parsing.

```
#include <stdlib.h>
```

```
#include <sys/types.h>
```

**Data Structures**

- [struct \\_mbus\\_frame](#)
- [struct \\_mbus\\_slave\\_data](#)
- [struct \\_mbus\\_data\\_information\\_block](#)

- struct [\\_mbus\\_value\\_information\\_block](#)
- struct [\\_mbus\\_data\\_record\\_header](#)
- struct [\\_mbus\\_data\\_record](#)
- struct [\\_mbus\\_data\\_variable\\_header](#)
- struct [\\_mbus\\_data\\_variable](#)
- struct [\\_mbus\\_data\\_fixed](#)
- struct [\\_mbus\\_frame\\_data](#)
- struct [\\_mbus\\_data\\_secondary\\_address](#)

## Defines

- #define [NITEMS\(x\)](#) (sizeof(x)/sizeof(x[0]))
- #define [MBUS\\_DIB\\_DIF\\_EXTENSION\\_BIT](#) 0x80
- #define [MBUS\\_DIB\\_VIF\\_EXTENSION\\_BIT](#) 0x80
- #define [MBUS\\_DATA\\_TYPE\\_FIXED](#) 1
- #define [MBUS\\_DATA\\_TYPE\\_VARIABLE](#) 2
- #define [MBUS\\_FRAME\\_TYPE\\_ANY](#) 0x00
- #define [MBUS\\_FRAME\\_TYPE\\_ACK](#) 0x01
- #define [MBUS\\_FRAME\\_TYPE\\_SHORT](#) 0x02
- #define [MBUS\\_FRAME\\_TYPE\\_CONTROL](#) 0x03
- #define [MBUS\\_FRAME\\_TYPE\\_LONG](#) 0x04
- #define [MBUS\\_FRAME\\_ACK\\_BASE\\_SIZE](#) 1
- #define [MBUS\\_FRAME\\_SHORT\\_BASE\\_SIZE](#) 5
- #define [MBUS\\_FRAME\\_CONTROL\\_BASE\\_SIZE](#) 9
- #define [MBUS\\_FRAME\\_LONG\\_BASE\\_SIZE](#) 9
- #define [MBUS\\_FRAME\\_BASE\\_SIZE\\_ACK](#) 1
- #define [MBUS\\_FRAME\\_BASE\\_SIZE\\_SHORT](#) 5
- #define [MBUS\\_FRAME\\_BASE\\_SIZE\\_CONTROL](#) 9
- #define [MBUS\\_FRAME\\_BASE\\_SIZE\\_LONG](#) 9
- #define [MBUS\\_FRAME\\_FIXED\\_SIZE\\_ACK](#) 1
- #define [MBUS\\_FRAME\\_FIXED\\_SIZE\\_SHORT](#) 5
- #define [MBUS\\_FRAME\\_FIXED\\_SIZE\\_CONTROL](#) 6
- #define [MBUS\\_FRAME\\_FIXED\\_SIZE\\_LONG](#) 6
- #define [MBUS\\_FRAME\\_ACK\\_START](#) 0xE5
- #define [MBUS\\_FRAME\\_SHORT\\_START](#) 0x10
- #define [MBUS\\_FRAME\\_CONTROL\\_START](#) 0x68
- #define [MBUS\\_FRAME\\_LONG\\_START](#) 0x68
- #define [MBUS\\_FRAME\\_STOP](#) 0x16
- #define [MBUS\\_MAX\\_PRIMARY\\_SLAVES](#) 256
- #define [MBUS\\_CONTROL\\_FIELD\\_DIRECTION](#) 0x07
- #define [MBUS\\_CONTROL\\_FIELD\\_FCB](#) 0x06
- #define [MBUS\\_CONTROL\\_FIELD\\_ACD](#) 0x06
- #define [MBUS\\_CONTROL\\_FIELD\\_FCV](#) 0x05
- #define [MBUS\\_CONTROL\\_FIELD\\_DFC](#) 0x05
- #define [MBUS\\_CONTROL\\_FIELD\\_F3](#) 0x04
- #define [MBUS\\_CONTROL\\_FIELD\\_F2](#) 0x03
- #define [MBUS\\_CONTROL\\_FIELD\\_F1](#) 0x02
- #define [MBUS\\_CONTROL\\_FIELD\\_F0](#) 0x01
- #define [MBUS\\_CONTROL\\_MASK\\_SND\\_NKE](#) 0x40
- #define [MBUS\\_CONTROL\\_MASK\\_SND\\_UD](#) 0x53

- #define MBUS\_CONTROL\_MASK\_REQ\_UD2 0x5B
- #define MBUS\_CONTROL\_MASK\_REQ\_UD1 0x5A
- #define MBUS\_CONTROL\_MASK\_RSP\_UD 0x08
- #define MBUS\_CONTROL\_MASK\_FCB 0x20
- #define MBUS\_CONTROL\_MASK\_FCV 0x10
- #define MBUS\_CONTROL\_MASK\_ACD 0x20
- #define MBUS\_CONTROL\_MASK\_DFC 0x10
- #define MBUS\_CONTROL\_MASK\_DIR 0x40
- #define MBUS\_CONTROL\_MASK\_DIR\_M2S 0x40
- #define MBUS\_CONTROL\_MASK\_DIR\_S2M 0x00
- #define MBUS\_ADDRESS\_BROADCAST\_REPLY 0xFE
- #define MBUS\_ADDRESS\_BROADCAST\_NOREPLY 0xFF
- #define MBUS\_ADDRESS\_NETWORK\_LAYER 0xFD
- #define MBUS\_CONTROL\_INFO\_DATA\_SEND 0x51
- #define MBUS\_CONTROL\_INFO\_DATA\_SEND\_MSB 0x55
- #define MBUS\_CONTROL\_INFO\_SELECT\_SLAVE 0x52
- #define MBUS\_CONTROL\_INFO\_SELECT\_SLAVE\_MSB 0x56
- #define MBUS\_CONTROL\_INFO\_APPLICATION\_RESET 0x50
- #define MBUS\_CONTROL\_INFO\_SYNC\_ACTION 0x54
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_300 0xB8
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_600 0xB9
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_1200 0xBA
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_2400 0xBB
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_4800 0xBC
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_9600 0xBD
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_19200 0xBE
- #define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_38400 0xBF
- #define MBUS\_CONTROL\_INFO\_REQUEST\_RAM\_READ 0xB1
- #define MBUS\_CONTROL\_INFO\_SEND\_USER\_DATA 0xB2
- #define MBUS\_CONTROL\_INFO\_INIT\_TEST\_CALIB 0xB3
- #define MBUS\_CONTROL\_INFO\_EEPROM\_READ 0xB4
- #define MBUS\_CONTROL\_INFO\_SW\_TEST\_START 0xB6
- #define MBUS\_CONTROL\_INFO\_RESP\_FIXED 0x73
- #define MBUS\_CONTROL\_INFO\_RESP\_FIXED\_MSB 0x77
- #define MBUS\_CONTROL\_INFO\_RESP\_VARIABLE 0x72
- #define MBUS\_CONTROL\_INFO\_RESP\_VARIABLE\_MSB 0x73
- #define MBUS\_DATA\_FIXED\_STATUS\_FORMAT\_MASK 0x80
- #define MBUS\_DATA\_FIXED\_STATUS\_FORMAT\_BCD 0x00
- #define MBUS\_DATA\_FIXED\_STATUS\_FORMAT\_INT 0x80
- #define MBUS\_DATA\_FIXED\_STATUS\_DATE\_MASK 0x40
- #define MBUS\_DATA\_FIXED\_STATUS\_DATE\_STORED 0x40
- #define MBUS\_DATA\_FIXED\_STATUS\_DATE\_CURRENT 0x00
- #define MBUS\_DATA\_RECORD\_DIF\_MASK\_INST 0x00
- #define MBUS\_DATA\_RECORD\_DIF\_MASK\_MIN 0x10
- #define MBUS\_DATA\_RECORD\_DIF\_MASK\_TYPE\_INT32 0x04
- #define MBUS\_DATA\_RECORD\_DIF\_MASK\_STORAGE\_NO 0x40
- #define MBUS\_DATA\_RECORD\_DIF\_MASK\_EXTENTION 0x80
- #define MBUS\_VARIABLE\_DATA\_MEDIUM\_OTHER 0x00
- #define MBUS\_VARIABLE\_DATA\_MEDIUM\_OIL 0x01
- #define MBUS\_VARIABLE\_DATA\_MEDIUM\_ELECTRICITY 0x02

- #define `MBUS_VARIABLE_DATA_MEDIUM_GAS` 0x03
- #define `MBUS_VARIABLE_DATA_MEDIUM_HEAT` 0x04
- #define `MBUS_VARIABLE_DATA_MEDIUM_STEAM` 0x05
- #define `MBUS_VARIABLE_DATA_MEDIUM_HOT_WATER` 0x06
- #define `MBUS_VARIABLE_DATA_MEDIUM_WATER` 0x07
- #define `MBUS_VARIABLE_DATA_MEDIUM_HEAT_COST` 0x08
- #define `MBUS_VARIABLE_DATA_MEDIUM_COMPR_AIR` 0x09
- #define `MBUS_VARIABLE_DATA_MEDIUM_COOL_OUT` 0x0A
- #define `MBUS_VARIABLE_DATA_MEDIUM_COOL_IN` 0x0B
- #define `MBUS_VARIABLE_DATA_MEDIUM_BUS` 0x0E
- #define `MBUS_VARIABLE_DATA_MEDIUM_COLD_WATER` 0x16
- #define `MBUS_VARIABLE_DATA_MEDIUM_DUAL_WATER` 0x17
- #define `MBUS_VARIABLE_DATA_MEDIUM_PRESSURE` 0x18
- #define `MBUS_VARIABLE_DATA_MEDIUM_ADC` 0x19

## Typedefs

- typedef struct `_mbus_frame` `mbus_frame`
- typedef struct `_mbus_slave_data` `mbus_slave_data`
- typedef struct `_mbus_data_information_block` `mbus_data_information_block`
- typedef struct `_mbus_value_information_block` `mbus_value_information_block`
- typedef struct `_mbus_data_record_header` `mbus_data_record_header`
- typedef struct `_mbus_data_record` `mbus_data_record`
- typedef struct `_mbus_data_variable_header` `mbus_data_variable_header`
- typedef struct `_mbus_data_variable` `mbus_data_variable`
- typedef struct `_mbus_data_fixed` `mbus_data_fixed`
- typedef struct `_mbus_frame_data` `mbus_frame_data`
- typedef struct `_mbus_data_secondary_address` `mbus_data_secondary_address`

## Functions

- `mbus_data_record * mbus_data_record_new ()`  
*Allocate and initialize a new variable data record.*
- `void mbus_data_record_free (mbus_data_record *record)`  
*free up memory associated with a data record and all the subsequent records in its list (apply recursively)*
- `void mbus_data_record_append (mbus_data_variable *data, mbus_data_record *record)`  
*Return a string containing an XML representation of the M-BUS frame.*
- `mbus_frame * mbus_frame_new (int frame_type)`  
*Allocate an M-bus frame data structure and initialize it according to which frame type is requested.*
- `int mbus_frame_free (mbus_frame *frame)`  
*Free the memory resources allocated for the M-Bus frame data structure.*
- `mbus_frame_data * mbus_frame_data_new ()`  
*Allocate and initialize a new frame data structure.*

- void `mbus_frame_data_free` (`mbus_frame_data` \*data)  
*Free up data associated with a frame data structure.*
- int `mbus_frame_calc_checksum` (`mbus_frame` \*frame)  
*Caclulate the checksum of the M-Bus frame.*
- int `mbus_frame_calc_length` (`mbus_frame` \*frame)  
*Calculate the values of the lengths fields in the M-Bus frame.*
- int `mbus_parse` (`mbus_frame` \*frame, `u_char` \*data, `size_t` data\_size)  
*PARSE M-BUS frame data structures from binary data.*
- int `mbus_data_fixed_parse` (`mbus_frame` \*frame, `mbus_data_fixed` \*data)  
*Parse the fixed-length data of a M-Bus frame.*
- int `mbus_data_variable_parse` (`mbus_frame` \*frame, `mbus_data_variable` \*data)  
*Parse the variable-length data of a M-Bus frame.*
- int `mbus_frame_data_parse` (`mbus_frame` \*frame, `mbus_frame_data` \*data)  
*Check the stype of the frame data (fixed or variable) and dispatch to the corresponding parser function.*
- int `mbus_frame_pack` (`mbus_frame` \*frame, `u_char` \*data, `size_t` data\_size)  
*Pack the M-bus frame into a binary string representation that can be sent on the bus.*
- int `mbus_frame_verify` (`mbus_frame` \*frame)  
*Verify that parsed frame is a valid M-bus frame.*
- int `mbus_frame_internal_pack` (`mbus_frame` \*frame, `mbus_frame_data` \*frame\_data)  
*pack the data stuctures into frame->data*
- const char \* `mbus_data_record_function` (`mbus_data_record` \*record)  
*Return a string containing the function description.*
- const char \* `mbus_data_fixed_function` (int status)  
*For fixed-length frames, return a string describing the type of value (stored or actual).*
- int `mbus_frame_type` (`mbus_frame` \*frame)  
*Return the M-Bus frame type.*
- `mbus_slave_data` \* `mbus_slave_data_get` (`size_t` i)  
*Return a pointer to the slave\_data register.*
- char \* `mbus_data_xml` (`mbus_frame_data` \*data)
- char \* `mbus_data_variable_xml` (`mbus_data_variable` \*data)  
*Generate XML for variable-length data.*
- char \* `mbus_data_fixed_xml` (`mbus_data_fixed` \*data)  
*Generate XML representation of fixed-length frame.*
- char \* `mbus_frame_data_xml` (`mbus_frame_data` \*data)

*Return a string containing an XML representation of the M-BUS frame.*

- char \* [mbus\\_data\\_variable\\_header\\_xml](#) (mbus\_data\_variable\_header \*header)  
*Generate XML for the variable-length data header.*
- int [mbus\\_frame\\_print](#) (mbus\_frame \*frame)  
*Dump frame in HEX on standard output.*
- int [mbus\\_frame\\_data\\_print](#) (mbus\_frame\_data \*data)  
*Print the data part of a frame.*
- int [mbus\\_data\\_fixed\\_print](#) (mbus\_data\_fixed \*data)
- int [mbus\\_data\\_variable\\_header\\_print](#) (mbus\_data\_variable\_header \*header)  
*Print M-bus frame info to stdout.*
- int [mbus\\_data\\_variable\\_print](#) (mbus\_data\_variable \*data)
- char \* [mbus\\_error\\_str](#) ()  
*Return a string that contains an the latest error message.*
- void [mbus\\_error\\_str\\_set](#) (char \*message)
- void [mbus\\_error\\_reset](#) ()
- int [mbus\\_data\\_manufacturer\\_encode](#) (u\_char \*m\_data, u\_char \*m\_code)  
*Generate manufacturer code from 2-byte encoded data.*
- const char \* [mbus\\_decode\\_manufacturer](#) (u\_char byte1, u\_char byte2)  
*Generate manufacturer code from 2-byte encoded data.*
- int [mbus\\_data\\_bcd\\_encode](#) (u\_char \*bcd\_data, size\_t bcd\_data\_size, int value)  
*Encode BCD data.*
- int [mbus\\_data\\_int\\_encode](#) (u\_char \*int\_data, size\_t int\_data\_size, int value)  
*Encode INTEGER data (into 'int\_data\_size' bytes).*
- long [mbus\\_data\\_bcd\\_decode](#) (u\_char \*bcd\_data, size\_t bcd\_data\_size)  
*Decode BCD data.*
- int [mbus\\_data\\_int\\_decode](#) (u\_char \*int\_data, size\_t int\_data\_size)  
*Decode INTEGER data.*
- long [mbus\\_data\\_long\\_decode](#) (u\_char \*int\_data, size\_t int\_data\_size)
- void [mbus\\_data\\_str\\_decode](#) (u\_char \*dst, const u\_char \*src, size\_t len)  
*Decode string data.*
- const char \* [mbus\\_data\\_fixed\\_medium](#) (mbus\_data\_fixed \*data)  
*For fixed-length frames, get a string describing the medium.*
- const char \* [mbus\\_data\\_fixed\\_unit](#) (int medium\_unit\_byte)  
*For fixed-length frames, get a string describing the unit of the data.*
- const char \* [mbus\\_data\\_variable\\_medium\\_lookup](#) (u\_char medium)

*For variable-length frames, returns a string describing the medium.*

- const char \* [mbus\\_unit\\_prefix](#) (int exp)

*Lookup the unit description from a VIF field in a data record.*

- const char \* [mbus\\_vib\\_unit\\_lookup](#) (mbus\_value\_information\_block \*vib)

*Lookup the unit from the VIB (VIF or VIFE).*

- const char \* [mbus\\_vif\\_unit\\_lookup](#) (u\_char vif)

*Look up the unit from a VIF field in the data record.*

- u\_char [mbus\\_dif\\_datalength\\_lookup](#) (u\_char dif)

*Look up the data length from a VIF field in the data record.*

- char \* [mbus\\_frame\\_get\\_secondary\\_address](#) (mbus\_frame \*frame)

- int [mbus\\_frame\\_select\\_secondary\\_pack](#) (mbus\_frame \*frame, char \*address)

### 5.4.1 Detailed Description

Functions and data structures for M-Bus protocol parsing.



## 5.4.2 Define Documentation

5.4.2.1 **#define MBUS\_ADDRESS\_BROADCAST\_NOREPLY 0xFF**

5.4.2.2 **#define MBUS\_ADDRESS\_BROADCAST\_REPLY 0xFE**

5.4.2.3 **#define MBUS\_ADDRESS\_NETWORK\_LAYER 0xFD**

5.4.2.4 **#define MBUS\_CONTROL\_FIELD\_ACD 0x06**

5.4.2.5 **#define MBUS\_CONTROL\_FIELD\_DFC 0x05**

5.4.2.6 **#define MBUS\_CONTROL\_FIELD\_DIRECTION 0x07**

5.4.2.7 **#define MBUS\_CONTROL\_FIELD\_F0 0x01**

5.4.2.8 **#define MBUS\_CONTROL\_FIELD\_F1 0x02**

5.4.2.9 **#define MBUS\_CONTROL\_FIELD\_F2 0x03**

5.4.2.10 **#define MBUS\_CONTROL\_FIELD\_F3 0x04**

5.4.2.11 **#define MBUS\_CONTROL\_FIELD\_FCB 0x06**

5.4.2.12 **#define MBUS\_CONTROL\_FIELD\_FCV 0x05**

5.4.2.13 **#define MBUS\_CONTROL\_INFO\_APPLICATION\_RESET 0x50**

5.4.2.14 **#define MBUS\_CONTROL\_INFO\_DATA\_SEND 0x51**

5.4.2.15 **#define MBUS\_CONTROL\_INFO\_DATA\_SEND\_MSB 0x55**

5.4.2.16 **#define MBUS\_CONTROL\_INFO\_EEPROM\_READ 0xB4**

5.4.2.17 **#define MBUS\_CONTROL\_INFO\_INIT\_TEST\_CALIB 0xB3**

5.4.2.18 **#define MBUS\_CONTROL\_INFO\_REQUEST\_RAM\_READ 0xB1**

5.4.2.19 **#define MBUS\_CONTROL\_INFO\_RESP\_FIXED 0x73**

5.4.2.20 **#define MBUS\_CONTROL\_INFO\_RESP\_FIXED\_MSB 0x77**

5.4.2.21 **#define MBUS\_CONTROL\_INFO\_RESP\_VARIABLE 0x72**

5.4.2.22 **#define MBUS\_CONTROL\_INFO\_RESP\_VARIABLE\_MSB 0x73**

5.4.2.23 **#define MBUS\_CONTROL\_INFO\_SELECT\_SLAVE 0x52**

5.4.2.24 **#define MBUS\_CONTROL\_INFO\_SELECT\_SLAVE\_MSB 0x56**

5.4.2.25 **#define MBUS\_CONTROL\_INFO\_SEND\_USER\_DATA 0xB2**

5.4.2.26 **#define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_1200 0xBA**

5.4.2.27 **#define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_19200 0xBB**

5.4.2.28 **#define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_2400 0xBB**

5.4.2.29 **#define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_300 0xB8**

5.4.2.30 **#define MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_38400 0xBF**

**5.4.4.2** `int mbus_data_bcd_encode ( u_char * bcd_data, size_t bcd_data_size, int value )`

Encode BCD data.

**5.4.4.3** `const char* mbus_data_fixed_function ( int status )`

For fixed-length frames, return a string describing the type of value (stored or actual).

**5.4.4.4** `const char* mbus_data_fixed_medium ( mbus_data_fixed * data )`

For fixed-length frames, get a string describing the medium.

**5.4.4.5** `int mbus_data_fixed_parse ( mbus_frame * frame, mbus_data_fixed * data )`

Parse the fixed-length data of a M-Bus frame.

**5.4.4.6** `int mbus_data_fixed_print ( mbus_data_fixed * data )`

**5.4.4.7** `const char* mbus_data_fixed_unit ( int medium_unit_byte )`

For fixed-length frames, get a string describing the unit of the data.

**5.4.4.8** `char* mbus_data_fixed_xml ( mbus_data_fixed * data )`

Generate XML representation of fixed-length frame.

**5.4.4.9** `int mbus_data_int_decode ( u_char * int_data, size_t int_data_size )`

Decode INTEGER data.

**5.4.4.10** `int mbus_data_int_encode ( u_char * int_data, size_t int_data_size, int value )`

Encode INTEGER data (into '*int\_data\_size*' bytes).

**5.4.4.11** `long mbus_data_long_decode ( u_char * int_data, size_t int_data_size )`

**5.4.4.12** `int mbus_data_manufacturer_encode ( u_char * m_data, u_char * m_code )`

Generate manufacturer code from 2-byte encoded data.

**5.4.4.13** `void mbus_data_record_append ( mbus_data_variable * data, mbus_data_record * record )`

Return a string containing an XML representation of the M-BUS frame.

**5.4.4.14** void `mbus_data_record_free` ( `mbus_data_record * record` )

free up memory associated with a data record and all the subsequent records in its list (apply recursively)

**5.4.4.15** const char\* `mbus_data_record_function` ( `mbus_data_record * record` )

Return a string containing the function description.

**5.4.4.16** `mbus_data_record*` `mbus_data_record_new` ( )

Allocate and initialize a new variable data record.

**5.4.4.17** void `mbus_data_str_decode` ( `u_char * dst`, `const u_char * src`, `size_t len` )

Decode string data.

**5.4.4.18** int `mbus_data_variable_header_print` ( `mbus_data_variable_header * header` )

Print M-bus frame info to stdout.

**5.4.4.19** char\* `mbus_data_variable_header_xml` ( `mbus_data_variable_header * header` )

Generate XML for the variable-length data header.

**5.4.4.20** const char\* `mbus_data_variable_medium_lookup` ( `u_char medium` )

For variable-length frames, returns a string describing the medium.

**5.4.4.21** int `mbus_data_variable_parse` ( `mbus_frame * frame`, `mbus_data_variable * data` )

Parse the variable-length data of a M-Bus frame.

**5.4.4.22** int `mbus_data_variable_print` ( `mbus_data_variable * data` )

**5.4.4.23** char\* `mbus_data_variable_xml` ( `mbus_data_variable * data` )

Generate XML for variable-length data.

**5.4.4.24** char\* `mbus_data_xml` ( `mbus_frame_data * data` )

**5.4.4.25** const char\* `mbus_decode_manufacturer` ( `u_char byte1`, `u_char byte2` )

Generate manufacturer code from 2-byte encoded data.

**5.4.4.26** `u_char mbus_dif_datalength_lookup ( u_char dif )`

Look up the data length from a VIF field in the data record.

See the table on page 41 the M-BUS specification.

**5.4.4.27** `void mbus_error_reset ( )`**5.4.4.28** `char* mbus_error_str ( )`

Return a string that contains an the latest error message.

**5.4.4.29** `void mbus_error_str_set ( char * message )`**5.4.4.30** `int mbus_frame_calc_checksum ( mbus_frame * frame )`

Calculate the checksum of the M-Bus frame.

The checksum algorithm is the arithmetic sum of the frame content, without using carry. Which content that is included in the checksum calculation depends on the frame type.

**5.4.4.31** `int mbus_frame_calc_length ( mbus_frame * frame )`

Calculate the values of the lengths fields in the M-Bus frame.

**5.4.4.32** `void mbus_frame_data_free ( mbus_frame_data * data )`

Free up data associated with a frame data structure.

**5.4.4.33** `mbus_frame_data* mbus_frame_data_new ( )`

Allocate and initialize a new frame data structure.

**5.4.4.34** `int mbus_frame_data_parse ( mbus_frame * frame, mbus_frame_data * data )`

Check the stype of the frame data (fixed or variable) and dispatch to the corresponding parser function.

**5.4.4.35** `int mbus_frame_data_print ( mbus_frame_data * data )`

Print the data part of a frame.

**5.4.4.36** `char* mbus_frame_data_xml ( mbus_frame_data * data )`

Return a string containing an XML representation of the M-BUS frame.

**5.4.4.37** `int mbus_frame_free ( mbus_frame * frame )`

Free the memory resources allocated for the M-Bus frame data structure.

**5.4.4.38** `char* mbus_frame_get_secondary_address ( mbus_frame * frame )`

**5.4.4.39** `int mbus_frame_internal_pack ( mbus_frame * frame, mbus_frame_data * frame_data )`

pack the data structures into frame->data

**5.4.4.40** `mbus_frame* mbus_frame_new ( int frame_type )`

Allocate an M-bus frame data structure and initialize it according to which frame type is requested.

**5.4.4.41** `int mbus_frame_pack ( mbus_frame * frame, u_char * data, size_t data_size )`

Pack the M-bus frame into a binary string representation that can be sent on the bus.

The binary packet format is different for the different types of M-bus frames.

**5.4.4.42** `int mbus_frame_print ( mbus_frame * frame )`

Dump frame in HEX on standard output.

**5.4.4.43** `int mbus_frame_select_secondary_pack ( mbus_frame * frame, char * address )`

**5.4.4.44** `int mbus_frame_type ( mbus_frame * frame )`

Return the M-Bus frame type.

**5.4.4.45** `int mbus_frame_verify ( mbus_frame * frame )`

Verify that parsed frame is a valid M-bus frame.

**5.4.4.46** `int mbus_parse ( mbus_frame * frame, u_char * data, size_t data_size )`

PARSE M-BUS frame data structures from binary data.

**5.4.4.47** `mbus_slave_data* mbus_slave_data_get ( size_t i )`

Return a pointer to the slave\_data register.

This register can be used for storing current slave status.

**5.4.4.48** `const char* mbus_unit_prefix ( int exp )`

Lookup the unit description from a VIF field in a data record.

**5.4.4.49** `const char* mbus_vib_unit_lookup ( mbus_value_information_block * vib )`

Lookup the unit from the VIB (VIF or VIFE).

#### 5.4.4.50 `const char* mbus_vif_unit_lookup ( u_char vif )`

Look up the unit from a VIF field in the data record.

See section 8.4.3 Codes for Value Information Field (VIF) in the M-BUS spec

## 5.5 mbus/mbus-serial.c File Reference

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/types.h>
#include <stdio.h>
#include <strings.h>
#include <termios.h>
#include <errno.h>
#include <string.h>
#include <mbus/mbus.h>
#include <mbus/mbus-serial.h>
```

### Defines

- `#define PACKET_BUFF_SIZE 2048`

### Functions

- `mbus_serial_handle * mbus_serial_connect (char *device)`  
*Set up a serial connection handle.*
- `int mbus_serial_disconnect (mbus_serial_handle *handle)`
- `int mbus_serial_send_frame (mbus_serial_handle *handle, mbus_frame *frame)`
- `int mbus_serial_rcv_frame (mbus_serial_handle *handle, mbus_frame *frame)`

### 5.5.1 Define Documentation

#### 5.5.1.1 `#define PACKET_BUFF_SIZE 2048`

### 5.5.2 Function Documentation

#### 5.5.2.1 `mbus_serial_handle* mbus_serial_connect ( char * device )`

Set up a serial connection handle.

**5.5.2.2** `int mbus_serial_disconnect ( mbus_serial_handle * handle )`

**5.5.2.3** `int mbus_serial_recv_frame ( mbus_serial_handle * handle, mbus_frame * frame )`

**5.5.2.4** `int mbus_serial_send_frame ( mbus_serial_handle * handle, mbus_frame * frame )`

## 5.6 mbus/mbus-serial.h File Reference

Functions and data structures for sending M-Bus data via RS232.

```
#include <termios.h>
#include <mbus/mbus.h>
```

### Data Structures

- [struct `\_mbus\_serial\_handle`](#)

### Typedefs

- [typedef struct `\_mbus\_serial\_handle` `mbus\_serial\_handle`](#)

### Functions

- [`mbus\_serial\_handle \* mbus\_serial\_connect` \(char \\*device\)](#)  
*Set up a serial connection handle.*
- [int `mbus\_serial\_disconnect` \(mbus\\_serial\\_handle \\*handle\)](#)
- [int `mbus\_serial\_send\_frame` \(mbus\\_serial\\_handle \\*handle, mbus\\_frame \\*frame\)](#)
- [int `mbus\_serial\_recv\_frame` \(mbus\\_serial\\_handle \\*handle, mbus\\_frame \\*frame\)](#)

#### 5.6.1 Detailed Description

Functions and data structures for sending M-Bus data via RS232.

#### 5.6.2 Typedef Documentation

**5.6.2.1** [typedef struct `\_mbus\_serial\_handle` `mbus\_serial\_handle`](#)

#### 5.6.3 Function Documentation

**5.6.3.1** [`mbus\_serial\_handle \* mbus\_serial\_connect` \( char \\* device \)](#)

Set up a serial connection handle.

**5.6.3.2** `int mbus_serial_disconnect ( mbus_serial_handle * handle )`

**5.6.3.3** `int mbus_serial_recv_frame ( mbus_serial_handle * handle, mbus_frame * frame )`

**5.6.3.4** `int mbus_serial_send_frame ( mbus_serial_handle * handle, mbus_frame * frame )`

## 5.7 mbus/mbus-tcp.c File Reference

```
#include <unistd.h>
#include <fcntl.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <strings.h>
#include <mbus/mbus.h>
#include <mbus/mbus-tcp.h>
```

### Defines

- #define [PACKET\\_BUFF\\_SIZE](#) 2048

### Functions

- [mbus\\_tcp\\_handle \\* mbus\\_tcp\\_connect](#) (char \*host, int port)  
*Setup a TCP/IP handle.*
- `int mbus_tcp_disconnect` (mbus\_tcp\_handle \*handle)
- `int mbus_tcp_send_frame` (mbus\_tcp\_handle \*handle, mbus\_frame \*frame)
- `int mbus_tcp_recv_frame` (mbus\_tcp\_handle \*handle, mbus\_frame \*frame)

### 5.7.1 Define Documentation

**5.7.1.1** #define [PACKET\\_BUFF\\_SIZE](#) 2048

### 5.7.2 Function Documentation

**5.7.2.1** `mbus_tcp_handle* mbus_tcp_connect ( char * host, int port )`

Setup a TCP/IP handle.

**5.7.2.2** `int mbus_tcp_disconnect ( mbus_tcp_handle * handle )`

**5.7.2.3** `int mbus_tcp_recv_frame ( mbus_tcp_handle * handle, mbus_frame * frame )`

**5.7.2.4** `int mbus_tcp_send_frame ( mbus_tcp_handle * handle, mbus_frame * frame )`

## 5.8 mbus/mbus-tcp.h File Reference

Functions and data structures for sending M-Bus data via TCP.

```
#include <mbus/mbus.h>
```

### Data Structures

- [struct \\_mbus\\_tcp\\_handle](#)

### Typedefs

- `typedef struct _mbus_tcp_handle mbus_tcp_handle`

### Functions

- `mbus_tcp_handle * mbus_tcp_connect (char *host, int port)`  
*Setup a TCP/IP handle.*
- `int mbus_tcp_disconnect (mbus_tcp_handle *handle)`
- `int mbus_tcp_send_frame (mbus_tcp_handle *handle, mbus_frame *frame)`
- `int mbus_tcp_recv_frame (mbus_tcp_handle *handle, mbus_frame *frame)`

#### 5.8.1 Detailed Description

Functions and data structures for sending M-Bus data via TCP.

#### 5.8.2 Typedef Documentation

**5.8.2.1** `typedef struct _mbus_tcp_handle mbus_tcp_handle`

#### 5.8.3 Function Documentation

**5.8.3.1** `mbus_tcp_handle* mbus_tcp_connect ( char * host, int port )`

Setup a TCP/IP handle.

**5.8.3.2** `int mbus_tcp_disconnect ( mbus_tcp_handle * handle )`

**5.8.3.3** `int mbus_tcp_recv_frame ( mbus_tcp_handle * handle, mbus_frame * frame )`

**5.8.3.4** `int mbus_tcp_send_frame ( mbus_tcp_handle * handle, mbus_frame * frame )`

## 5.9 mbus/mbus.c File Reference

```
#include <mbus/mbus-protocol.h>
```

### Functions

- `int mbus_init ()`

### 5.9.1 Function Documentation

**5.9.1.1** `int mbus_init ( )`

## 5.10 mbus/mbus.h File Reference

Main include file for the Freescada libmbus library.

```
#include <mbus/mbus-protocol.h>
```

```
#include <mbus/mbus-protocol-aux.h>
```

```
#include <mbus/mbus-tcp.h>
```

```
#include <mbus/mbus-serial.h>
```

### Functions

- `int mbus_init ()`

### 5.10.1 Detailed Description

Main include file for the Freescada libmbus library. Include this file to access the libmbus API:

```
#include <mbus/mbus.h>
```

### 5.10.2 Function Documentation

**5.10.2.1** `int mbus_init ( )`

# Index

- [\\_mbus\\_address](#), 7
  - [is\\_primary](#), 7
  - [primary](#), 7
  - [secondary](#), 7
- [\\_mbus\\_data\\_fixed](#), 8
  - [cnt1\\_type](#), 8
  - [cnt1\\_val](#), 8
  - [cnt2\\_type](#), 8
  - [cnt2\\_val](#), 8
  - [id\\_bcd](#), 8
  - [status](#), 8
  - [tx\\_cnt](#), 8
- [\\_mbus\\_data\\_information\\_block](#), 8
  - [dif](#), 9
  - [dife](#), 9
  - [ndife](#), 9
- [\\_mbus\\_data\\_record](#), 9
  - [data](#), 9
  - [data\\_len](#), 9
  - [drh](#), 9
  - [next](#), 9
- [\\_mbus\\_data\\_record\\_header](#), 9
  - [dib](#), 10
  - [vib](#), 10
- [\\_mbus\\_data\\_secondary\\_address](#), 10
  - [id\\_bcd](#), 10
  - [manufacturer](#), 10
  - [medium](#), 10
  - [version](#), 10
- [\\_mbus\\_data\\_variable](#), 10
  - [data](#), 11
  - [data\\_len](#), 11
  - [header](#), 11
  - [mdh](#), 11
  - [mfg\\_data](#), 11
  - [mfg\\_data\\_len](#), 11
  - [nrecords](#), 11
  - [record](#), 11
- [\\_mbus\\_data\\_variable\\_header](#), 11
  - [access\\_no](#), 12
  - [id\\_bcd](#), 12
  - [manufacturer](#), 12
  - [medium](#), 12
  - [signature](#), 12
  - [status](#), 12
  - [version](#), 12
- [\\_mbus\\_frame](#), 12
  - [address](#), 13
  - [checksum](#), 13
  - [control](#), 13
  - [control\\_information](#), 13
  - [data](#), 13
  - [data\\_size](#), 13
  - [length1](#), 13
  - [length2](#), 13
  - [start1](#), 13
  - [start2](#), 13
  - [stop](#), 13
  - [type](#), 13
- [\\_mbus\\_frame\\_data](#), 13
  - [data\\_fix](#), 13
  - [data\\_var](#), 13
  - [type](#), 13
- [\\_mbus\\_handle](#), 14
  - [is\\_serial](#), 14
  - [m\\_serial\\_handle](#), 14
  - [m\\_tcp\\_handle](#), 14
- [\\_mbus\\_record](#), 15
  - [function\\_medium](#), 15
  - [is\\_numeric](#), 15
  - [quantity](#), 15
  - [unit](#), 15
  - [value](#), 15
- [\\_mbus\\_serial\\_handle](#), 16
  - [device](#), 16
  - [fd](#), 16
  - [t](#), 16
- [\\_mbus\\_slave\\_data](#), 16
  - [state\\_acd](#), 16
  - [state\\_fcb](#), 16
- [\\_mbus\\_string](#), 17
  - [size](#), 17
  - [value](#), 17
- [\\_mbus\\_tcp\\_handle](#), 17
  - [host](#), 18
  - [port](#), 18
  - [sock](#), 18
- [\\_mbus\\_value](#), 18
  - [real\\_val](#), 18
  - [str\\_val](#), 18



- mbus\_read\_slave, 24
- mbus\_record\_free, 25
- mbus\_record\_new, 25
- mbus\_rcv\_frame, 25
- mbus\_scan\_2nd\_address\_range, 25
- mbus\_send\_frame, 26
- mbus\_send\_ping\_frame, 26
- mbus\_send\_request\_frame, 26
- mbus\_send\_select\_frame, 26
- mbus\_variable\_value\_decode, 26
- mbus\_variable\_vif, 23
- mbus\_vib\_unit\_normalize, 26
- mbus\_vif\_unit\_normalize, 27
- vif\_table, 27
- mbus-protocol-aux.h
  - mbus\_address, 30
  - mbus\_connect\_serial, 31
  - mbus\_connect\_tcp, 31
  - mbus\_data\_fixed\_normalize, 31
  - mbus\_data\_variable\_value\_decode, 32
  - mbus\_disconnect, 32
  - mbus\_handle, 30
  - mbus\_parse\_fixed\_record, 32
  - mbus\_parse\_variable\_record, 33
  - MBUS\_PROBE\_COLLISION, 30
  - MBUS\_PROBE\_ERROR, 30
  - MBUS\_PROBE\_NOTHING, 30
  - mbus\_probe\_secondary\_address, 33
  - MBUS\_PROBE\_SINGLE, 30
  - mbus\_read\_slave, 33
  - mbus\_record, 31
  - mbus\_record\_free, 34
  - mbus\_record\_new, 34
  - mbus\_rcv\_frame, 34
  - mbus\_scan\_2nd\_address\_range, 34
  - mbus\_send\_data\_request\_frame, 34
  - mbus\_send\_frame, 35
  - mbus\_send\_select\_frame, 35
  - mbus\_string, 31
  - mbus\_value, 31
  - mbus\_vib\_unit\_normalize, 35
  - mbus\_vif\_unit\_normalize, 35
- mbus-protocol.c
  - calc\_checksum, 39
  - calc\_length, 39
  - error\_str, 44
  - mbus\_data\_bcd\_decode, 40
  - mbus\_data\_bcd\_encode, 40
  - mbus\_data\_fixed\_function, 40
  - mbus\_data\_fixed\_medium, 40
  - mbus\_data\_fixed\_parse, 40
  - mbus\_data\_fixed\_print, 40
  - mbus\_data\_fixed\_unit, 40
  - mbus\_data\_fixed\_xml, 40
  - mbus\_data\_int\_decode, 40
  - mbus\_data\_int\_encode, 40
  - mbus\_data\_long\_decode, 40
  - mbus\_data\_manufacturer\_encode, 41
  - mbus\_data\_record\_append, 41
  - mbus\_data\_record\_decode, 41
  - mbus\_data\_record\_free, 41
  - mbus\_data\_record\_function, 41
  - mbus\_data\_record\_new, 41
  - mbus\_data\_record\_unit, 41
  - mbus\_data\_record\_value, 41
  - mbus\_data\_str\_decode, 41
  - mbus\_data\_variable\_header\_print, 41
  - mbus\_data\_variable\_header\_xml, 41
  - mbus\_data\_variable\_medium\_lookup, 41
  - mbus\_data\_variable\_parse, 42
  - mbus\_data\_variable\_print, 42
  - mbus\_data\_variable\_xml, 42
  - mbus\_decode\_manufacturer, 42
  - mbus\_dif\_datalength\_lookup, 42
  - mbus\_error\_reset, 42
  - mbus\_error\_str, 42
  - mbus\_error\_str\_set, 42
  - mbus\_frame\_calc\_checksum, 42
  - mbus\_frame\_calc\_length, 42
  - mbus\_frame\_data\_free, 42
  - mbus\_frame\_data\_new, 42
  - mbus\_frame\_data\_parse, 43
  - mbus\_frame\_data\_print, 43
  - mbus\_frame\_data\_xml, 43
  - mbus\_frame\_free, 43
  - mbus\_frame\_get\_secondary\_address, 43
  - mbus\_frame\_internal\_pack, 43
  - mbus\_frame\_new, 43
  - mbus\_frame\_pack, 43
  - mbus\_frame\_print, 43
  - mbus\_frame\_select\_secondary\_pack, 43
  - mbus\_frame\_type, 43
  - mbus\_frame\_verify, 43
  - mbus\_parse, 44
  - mbus\_slave\_data\_get, 44
  - mbus\_unit\_prefix, 44
  - mbus\_vib\_unit\_lookup, 44
  - mbus\_vif\_unit\_lookup, 44
  - NITEMS, 39
  - parse\_debug, 44
  - slave\_data, 44
- mbus-protocol.h
  - MBUS\_ADDRESS\_BROADCAST\_-  
NOREPLY, 52
  - MBUS\_ADDRESS\_BROADCAST\_REPLY,  
52
  - MBUS\_ADDRESS\_NETWORK\_LAYER, 52
  - MBUS\_CONTROL\_FIELD\_ACD, 52

- MBUS\_CONTROL\_FIELD\_DFC, [52](#)
- MBUS\_CONTROL\_FIELD\_DIRECTION, [52](#)
- MBUS\_CONTROL\_FIELD\_F0, [52](#)
- MBUS\_CONTROL\_FIELD\_F1, [52](#)
- MBUS\_CONTROL\_FIELD\_F2, [52](#)
- MBUS\_CONTROL\_FIELD\_F3, [52](#)
- MBUS\_CONTROL\_FIELD\_FCB, [52](#)
- MBUS\_CONTROL\_FIELD\_FCV, [52](#)
- MBUS\_CONTROL\_INFO\_APPLICATION\_-  
RESET, [52](#)
- MBUS\_CONTROL\_INFO\_DATA\_SEND, [52](#)
- MBUS\_CONTROL\_INFO\_DATA\_SEND\_-  
MSB, [52](#)
- MBUS\_CONTROL\_INFO\_EEPROM\_-  
READ, [52](#)
- MBUS\_CONTROL\_INFO\_INIT\_TEST\_-  
CALIB, [52](#)
- MBUS\_CONTROL\_INFO\_REQUEST\_-  
RAM\_READ, [52](#)
- MBUS\_CONTROL\_INFO\_RESP\_FIXED, [52](#)
- MBUS\_CONTROL\_INFO\_RESP\_FIXED\_-  
MSB, [52](#)
- MBUS\_CONTROL\_INFO\_RESP\_-  
VARIABLE, [52](#)
- MBUS\_CONTROL\_INFO\_RESP\_-  
VARIABLE\_MSB, [52](#)
- MBUS\_CONTROL\_INFO\_SELECT\_-  
SLAVE, [52](#)
- MBUS\_CONTROL\_INFO\_SELECT\_-  
SLAVE\_MSB, [52](#)
- MBUS\_CONTROL\_INFO\_SEND\_USER\_-  
DATA, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_1200, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_19200, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_2400, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_300, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_38400, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_4800, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_600, [52](#)
- MBUS\_CONTROL\_INFO\_SET\_-  
BAUDRATE\_9600, [52](#)
- MBUS\_CONTROL\_INFO\_SW\_TEST\_-  
START, [52](#)
- MBUS\_CONTROL\_INFO\_SYNC\_ACTION,  
[52](#)
- MBUS\_CONTROL\_MASK\_ACD, [52](#)
- MBUS\_CONTROL\_MASK\_DFC, [52](#)
- MBUS\_CONTROL\_MASK\_DIR, [52](#)
- MBUS\_CONTROL\_MASK\_DIR\_M2S, [52](#)
- MBUS\_CONTROL\_MASK\_DIR\_S2M, [52](#)
- MBUS\_CONTROL\_MASK\_FCB, [52](#)
- MBUS\_CONTROL\_MASK\_FCV, [52](#)
- MBUS\_CONTROL\_MASK\_REQ\_UD1, [52](#)
- MBUS\_CONTROL\_MASK\_REQ\_UD2, [52](#)
- MBUS\_CONTROL\_MASK\_RSP\_UD, [52](#)
- MBUS\_CONTROL\_MASK\_SND\_NKE, [52](#)
- MBUS\_CONTROL\_MASK\_SND\_UD, [52](#)
- mbus\_data\_bcd\_decode, [52](#)
- mbus\_data\_bcd\_encode, [52](#)
- mbus\_data\_fixed, [52](#)
- mbus\_data\_fixed\_function, [53](#)
- mbus\_data\_fixed\_medium, [53](#)
- mbus\_data\_fixed\_parse, [53](#)
- mbus\_data\_fixed\_print, [53](#)
- MBUS\_DATA\_FIXED\_STATUS\_DATE\_-  
CURRENT, [52](#)
- MBUS\_DATA\_FIXED\_STATUS\_DATE\_-  
MASK, [52](#)
- MBUS\_DATA\_FIXED\_STATUS\_DATE\_-  
STORED, [52](#)
- MBUS\_DATA\_FIXED\_STATUS\_-  
FORMAT\_BCD, [52](#)
- MBUS\_DATA\_FIXED\_STATUS\_-  
FORMAT\_INT, [52](#)
- MBUS\_DATA\_FIXED\_STATUS\_-  
FORMAT\_MASK, [52](#)
- mbus\_data\_fixed\_unit, [53](#)
- mbus\_data\_fixed\_xml, [53](#)
- mbus\_data\_information\_block, [52](#)
- mbus\_data\_int\_decode, [53](#)
- mbus\_data\_int\_encode, [53](#)
- mbus\_data\_long\_decode, [53](#)
- mbus\_data\_manufacturer\_encode, [53](#)
- mbus\_data\_record, [52](#)
- mbus\_data\_record\_append, [53](#)
- MBUS\_DATA\_RECORD\_DIF\_MASK\_-  
EXTENTION, [52](#)
- MBUS\_DATA\_RECORD\_DIF\_MASK\_-  
INST, [52](#)
- MBUS\_DATA\_RECORD\_DIF\_MASK\_MIN,  
[52](#)
- MBUS\_DATA\_RECORD\_DIF\_MASK\_-  
STORAGE\_NO, [52](#)
- MBUS\_DATA\_RECORD\_DIF\_MASK\_-  
TYPE\_INT32, [52](#)
- mbus\_data\_record\_free, [53](#)
- mbus\_data\_record\_function, [54](#)
- mbus\_data\_record\_header, [52](#)
- mbus\_data\_record\_new, [54](#)
- mbus\_data\_secondary\_address, [52](#)
- mbus\_data\_str\_decode, [54](#)

- MBUS\_DATA\_TYPE\_FIXED, 52
- MBUS\_DATA\_TYPE\_VARIABLE, 52
- mbus\_data\_variable, 52
- mbus\_data\_variable\_header, 52
- mbus\_data\_variable\_header\_print, 54
- mbus\_data\_variable\_header\_xml, 54
- mbus\_data\_variable\_medium\_lookup, 54
- mbus\_data\_variable\_parse, 54
- mbus\_data\_variable\_print, 54
- mbus\_data\_variable\_xml, 54
- mbus\_data\_xml, 54
- mbus\_decode\_manufacturer, 54
- MBUS\_DIB\_DIF\_EXTENSION\_BIT, 52
- MBUS\_DIB\_VIF\_EXTENSION\_BIT, 52
- mbus\_dif\_datalength\_lookup, 54
- mbus\_error\_reset, 55
- mbus\_error\_str, 55
- mbus\_error\_str\_set, 55
- mbus\_frame, 52
- MBUS\_FRAME\_ACK\_BASE\_SIZE, 52
- MBUS\_FRAME\_ACK\_START, 52
- MBUS\_FRAME\_BASE\_SIZE\_ACK, 52
- MBUS\_FRAME\_BASE\_SIZE\_CONTROL, 52
- MBUS\_FRAME\_BASE\_SIZE\_LONG, 52
- MBUS\_FRAME\_BASE\_SIZE\_SHORT, 52
- mbus\_frame\_calc\_checksum, 55
- mbus\_frame\_calc\_length, 55
- MBUS\_FRAME\_CONTROL\_BASE\_SIZE, 52
- MBUS\_FRAME\_CONTROL\_START, 52
- mbus\_frame\_data, 52
- mbus\_frame\_data\_free, 55
- mbus\_frame\_data\_new, 55
- mbus\_frame\_data\_parse, 55
- mbus\_frame\_data\_print, 55
- mbus\_frame\_data\_xml, 55
- MBUS\_FRAME\_FIXED\_SIZE\_ACK, 52
- MBUS\_FRAME\_FIXED\_SIZE\_CONTROL, 52
- MBUS\_FRAME\_FIXED\_SIZE\_LONG, 52
- MBUS\_FRAME\_FIXED\_SIZE\_SHORT, 52
- mbus\_frame\_free, 55
- mbus\_frame\_get\_secondary\_address, 55
- mbus\_frame\_internal\_pack, 56
- MBUS\_FRAME\_LONG\_LONG\_BASE\_SIZE, 52
- MBUS\_FRAME\_LONG\_LONG\_START, 52
- mbus\_frame\_new, 56
- mbus\_frame\_pack, 56
- mbus\_frame\_print, 56
- mbus\_frame\_select\_secondary\_pack, 56
- MBUS\_FRAME\_SHORT\_BASE\_SIZE, 52
- MBUS\_FRAME\_SHORT\_START, 52
- MBUS\_FRAME\_STOP, 52
- mbus\_frame\_type, 56
- MBUS\_FRAME\_TYPE\_ACK, 52
- MBUS\_FRAME\_TYPE\_ANY, 52
- MBUS\_FRAME\_TYPE\_CONTROL, 52
- MBUS\_FRAME\_TYPE\_LONG, 52
- MBUS\_FRAME\_TYPE\_SHORT, 52
- mbus\_frame\_verify, 56
- MBUS\_MAX\_PRIMARY\_SLAVES, 52
- mbus\_parse, 56
- mbus\_slave\_data, 52
- mbus\_slave\_data\_get, 56
- mbus\_unit\_prefix, 56
- mbus\_value\_information\_block, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
ADC, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_BUS, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
COLD\_WATER, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
COMPR\_AIR, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
COOL\_IN, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
COOL\_OUT, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
DUAL\_WATER, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
ELECTRICITY, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_GAS, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
HEAT, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
HEAT\_COST, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
HOT\_WATER, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_OIL, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
OTHER, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
PRESSURE, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
STEAM, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
WATER, 52
- mbus\_vib\_unit\_lookup, 56
- mbus\_vif\_unit\_lookup, 56
- NITEMS, 52
- mbus-serial.c
  - mbus\_serial\_connect, 57
  - mbus\_serial\_disconnect, 57
  - mbus\_serial\_recv\_frame, 58

- mbus\_serial\_send\_frame, 58
  - PACKET\_BUFF\_SIZE, 57
- mbus-serial.h
  - mbus\_serial\_connect, 58
  - mbus\_serial\_disconnect, 58
  - mbus\_serial\_handle, 58
  - mbus\_serial\_recv\_frame, 59
  - mbus\_serial\_send\_frame, 59
- mbus-tcp.c
  - mbus\_tcp\_connect, 59
  - mbus\_tcp\_disconnect, 59
  - mbus\_tcp\_recv\_frame, 60
  - mbus\_tcp\_send\_frame, 60
  - PACKET\_BUFF\_SIZE, 59
- mbus-tcp.h
  - mbus\_tcp\_connect, 60
  - mbus\_tcp\_disconnect, 60
  - mbus\_tcp\_handle, 60
  - mbus\_tcp\_recv\_frame, 61
  - mbus\_tcp\_send\_frame, 61
- mbus.c
  - mbus\_init, 61
- mbus.h
  - mbus\_init, 61
- mbus/mbus-protocol-aux.c, 21
- mbus/mbus-protocol-aux.h, 27
- mbus/mbus-protocol.c, 36
- mbus/mbus-protocol.h, 44
- mbus/mbus-serial.c, 57
- mbus/mbus-serial.h, 58
- mbus/mbus-tcp.c, 59
- mbus/mbus-tcp.h, 60
- mbus/mbus.c, 61
- mbus/mbus.h, 61
- mbus\_address
  - mbus-protocol-aux.h, 30
- MBUS\_ADDRESS\_BROADCAST\_NOREPLY
  - mbus-protocol.h, 52
- MBUS\_ADDRESS\_BROADCAST\_REPLY
  - mbus-protocol.h, 52
- MBUS\_ADDRESS\_NETWORK\_LAYER
  - mbus-protocol.h, 52
- mbus\_connect\_serial
  - mbus-protocol-aux.c, 23
  - mbus-protocol-aux.h, 31
- mbus\_connect\_tcp
  - mbus-protocol-aux.c, 23
  - mbus-protocol-aux.h, 31
- MBUS\_CONTROL\_FIELD\_ACD
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_DFC
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_DIRECTION
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_F0
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_F1
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_F2
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_F3
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_FCB
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_FIELD\_FCV
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_APPLICATION\_RESET
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_DATA\_SEND
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_DATA\_SEND\_MSB
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_EEPROM\_READ
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_INIT\_TEST\_CALIB
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_REQUEST\_RAM\_READ
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_RESP\_FIXED
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_RESP\_FIXED\_MSB
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_RESP\_VARIABLE
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_RESP\_VARIABLE\_MSB
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SELECT\_SLAVE
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SELECT\_SLAVE\_MSB
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SEND\_USER\_DATA
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_1200
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_19200
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_2400
  - mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_300
  - mbus-protocol.h, 52

- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_-  
38400  
mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_-  
4800  
mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_-  
600  
mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SET\_BAUDRATE\_-  
9600  
mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SW\_TEST\_START  
mbus-protocol.h, 52
- MBUS\_CONTROL\_INFO\_SYNC\_ACTION  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_ACD  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_DFC  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_DIR  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_DIR\_M2S  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_DIR\_S2M  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_FCB  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_FCV  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_REQ\_UD1  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_REQ\_UD2  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_RSP\_UD  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_SND\_NKE  
mbus-protocol.h, 52
- MBUS\_CONTROL\_MASK\_SND\_UD  
mbus-protocol.h, 52
- mbus\_data\_bcd\_decode  
mbus-protocol.c, 40  
mbus-protocol.h, 52
- mbus\_data\_bcd\_encode  
mbus-protocol.c, 40  
mbus-protocol.h, 52
- mbus\_data\_fixed  
mbus-protocol.h, 52
- mbus\_data\_fixed\_function  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_fixed\_medium  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_fixed\_normalize  
mbus-protocol-aux.h, 31
- mbus\_data\_fixed\_parse  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_fixed\_print  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- MBUS\_DATA\_FIXED\_STATUS\_DATE\_-  
CURRENT  
mbus-protocol.h, 52
- MBUS\_DATA\_FIXED\_STATUS\_DATE\_MASK  
mbus-protocol.h, 52
- MBUS\_DATA\_FIXED\_STATUS\_DATE\_-  
STORED  
mbus-protocol.h, 52
- MBUS\_DATA\_FIXED\_STATUS\_FORMAT\_BCD  
mbus-protocol.h, 52
- MBUS\_DATA\_FIXED\_STATUS\_FORMAT\_INT  
mbus-protocol.h, 52
- MBUS\_DATA\_FIXED\_STATUS\_FORMAT\_-  
MASK  
mbus-protocol.h, 52
- mbus\_data\_fixed\_unit  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_fixed\_xml  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_information\_block  
mbus-protocol.h, 52
- mbus\_data\_int\_decode  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_int\_encode  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_long\_decode  
mbus-protocol.c, 40  
mbus-protocol.h, 53
- mbus\_data\_manufacturer\_encode  
mbus-protocol.c, 41  
mbus-protocol.h, 53
- mbus\_data\_record  
mbus-protocol.h, 52
- mbus\_data\_record\_append  
mbus-protocol.c, 41  
mbus-protocol.h, 53
- mbus\_data\_record\_decode  
mbus-protocol.c, 41
- MBUS\_DATA\_RECORD\_DIF\_MASK\_-  
EXTENTION  
mbus-protocol.h, 52
- MBUS\_DATA\_RECORD\_DIF\_MASK\_INST

- mbus-protocol.h, 52
- MBUS\_DATA\_RECORD\_DIF\_MASK\_MIN
  - mbus-protocol.h, 52
- MBUS\_DATA\_RECORD\_DIF\_MASK\_-  
STORAGE\_NO
  - mbus-protocol.h, 52
- MBUS\_DATA\_RECORD\_DIF\_MASK\_TYPE\_-  
INT32
  - mbus-protocol.h, 52
- mbus\_data\_record\_free
  - mbus-protocol.c, 41
  - mbus-protocol.h, 53
- mbus\_data\_record\_function
  - mbus-protocol.c, 41
  - mbus-protocol.h, 54
- mbus\_data\_record\_header
  - mbus-protocol.h, 52
- mbus\_data\_record\_new
  - mbus-protocol.c, 41
  - mbus-protocol.h, 54
- mbus\_data\_record\_unit
  - mbus-protocol.c, 41
- mbus\_data\_record\_value
  - mbus-protocol.c, 41
- mbus\_data\_secondary\_address
  - mbus-protocol.h, 52
- mbus\_data\_str\_decode
  - mbus-protocol.c, 41
  - mbus-protocol.h, 54
- MBUS\_DATA\_TYPE\_FIXED
  - mbus-protocol.h, 52
- MBUS\_DATA\_TYPE\_VARIABLE
  - mbus-protocol.h, 52
- mbus\_data\_variable
  - mbus-protocol.h, 52
- mbus\_data\_variable\_header
  - mbus-protocol.h, 52
- mbus\_data\_variable\_header\_print
  - mbus-protocol.c, 41
  - mbus-protocol.h, 54
- mbus\_data\_variable\_header\_xml
  - mbus-protocol.c, 41
  - mbus-protocol.h, 54
- mbus\_data\_variable\_medium\_lookup
  - mbus-protocol.c, 41
  - mbus-protocol.h, 54
- mbus\_data\_variable\_parse
  - mbus-protocol.c, 42
  - mbus-protocol.h, 54
- mbus\_data\_variable\_print
  - mbus-protocol.c, 42
  - mbus-protocol.h, 54
- mbus\_data\_variable\_value\_decode
  - mbus-protocol-aux.h, 32
- mbus\_data\_variable\_xml
  - mbus-protocol.c, 42
  - mbus-protocol.h, 54
- mbus\_data\_xml
  - mbus-protocol.h, 54
- MBUS\_DEBUG
  - mbus-protocol-aux.c, 23
- mbus\_decode\_manufacturer
  - mbus-protocol.c, 42
  - mbus-protocol.h, 54
- MBUS\_DIB\_DIF\_EXTENSION\_BIT
  - mbus-protocol.h, 52
- MBUS\_DIB\_VIF\_EXTENSION\_BIT
  - mbus-protocol.h, 52
- mbus\_dif\_data\_length\_lookup
  - mbus-protocol.c, 42
  - mbus-protocol.h, 54
- mbus\_disconnect
  - mbus-protocol-aux.c, 23
  - mbus-protocol-aux.h, 32
- MBUS\_ERROR
  - mbus-protocol-aux.c, 23
- mbus\_error\_reset
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- mbus\_error\_str
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- mbus\_error\_str\_set
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- mbus\_fixed\_normalize
  - mbus-protocol-aux.c, 23
- mbus\_frame
  - mbus-protocol.h, 52
- MBUS\_FRAME\_ACK\_BASE\_SIZE
  - mbus-protocol.h, 52
- MBUS\_FRAME\_ACK\_START
  - mbus-protocol.h, 52
- MBUS\_FRAME\_BASE\_SIZE\_ACK
  - mbus-protocol.h, 52
- MBUS\_FRAME\_BASE\_SIZE\_CONTROL
  - mbus-protocol.h, 52
- MBUS\_FRAME\_BASE\_SIZE\_LONG
  - mbus-protocol.h, 52
- MBUS\_FRAME\_BASE\_SIZE\_SHORT
  - mbus-protocol.h, 52
- mbus\_frame\_calc\_checksum
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- mbus\_frame\_calc\_length
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- MBUS\_FRAME\_CONTROL\_BASE\_SIZE

- mbus-protocol.h, 52
- MBUS\_FRAME\_CONTROL\_START
  - mbus-protocol.h, 52
- mbus\_frame\_data
  - mbus-protocol.h, 52
- mbus\_frame\_data\_free
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- mbus\_frame\_data\_new
  - mbus-protocol.c, 42
  - mbus-protocol.h, 55
- mbus\_frame\_data\_parse
  - mbus-protocol.c, 43
  - mbus-protocol.h, 55
- mbus\_frame\_data\_print
  - mbus-protocol.c, 43
  - mbus-protocol.h, 55
- mbus\_frame\_data\_xml
  - mbus-protocol.c, 43
  - mbus-protocol.h, 55
- MBUS\_FRAME\_FIXED\_SIZE\_ACK
  - mbus-protocol.h, 52
- MBUS\_FRAME\_FIXED\_SIZE\_CONTROL
  - mbus-protocol.h, 52
- MBUS\_FRAME\_FIXED\_SIZE\_LONG
  - mbus-protocol.h, 52
- MBUS\_FRAME\_FIXED\_SIZE\_SHORT
  - mbus-protocol.h, 52
- mbus\_frame\_free
  - mbus-protocol.c, 43
  - mbus-protocol.h, 55
- mbus\_frame\_get\_secondary\_address
  - mbus-protocol.c, 43
  - mbus-protocol.h, 55
- mbus\_frame\_internal\_pack
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- MBUS\_FRAME\_LONG\_BASE\_SIZE
  - mbus-protocol.h, 52
- MBUS\_FRAME\_LONG\_START
  - mbus-protocol.h, 52
- mbus\_frame\_new
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- mbus\_frame\_pack
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- mbus\_frame\_print
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- mbus\_frame\_select\_secondary\_pack
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- MBUS\_FRAME\_SHORT\_BASE\_SIZE
  - mbus-protocol.h, 52
- MBUS\_FRAME\_SHORT\_START
  - mbus-protocol.h, 52
- MBUS\_FRAME\_STOP
  - mbus-protocol.h, 52
- mbus\_frame\_type
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- MBUS\_FRAME\_TYPE\_ACK
  - mbus-protocol.h, 52
- MBUS\_FRAME\_TYPE\_ANY
  - mbus-protocol.h, 52
- MBUS\_FRAME\_TYPE\_CONTROL
  - mbus-protocol.h, 52
- MBUS\_FRAME\_TYPE\_LONG
  - mbus-protocol.h, 52
- MBUS\_FRAME\_TYPE\_SHORT
  - mbus-protocol.h, 52
- mbus\_frame\_verify
  - mbus-protocol.c, 43
  - mbus-protocol.h, 56
- mbus\_handle
  - mbus-protocol-aux.h, 30
- mbus\_init
  - mbus.c, 61
  - mbus.h, 61
- MBUS\_MAX\_PRIMARY\_SLAVES
  - mbus-protocol.h, 52
- mbus\_parse
  - mbus-protocol.c, 44
  - mbus-protocol.h, 56
- mbus\_parse\_fixed\_record
  - mbus-protocol-aux.c, 24
  - mbus-protocol-aux.h, 32
- mbus\_parse\_variable\_record
  - mbus-protocol-aux.c, 24
  - mbus-protocol-aux.h, 33
- MBUS\_PROBE\_COLLISION
  - mbus-protocol-aux.h, 30
- MBUS\_PROBE\_ERROR
  - mbus-protocol-aux.h, 30
- MBUS\_PROBE\_NOTHING
  - mbus-protocol-aux.h, 30
- mbus\_probe\_secondary\_address
  - mbus-protocol-aux.c, 24
  - mbus-protocol-aux.h, 33
- MBUS\_PROBE\_SINGLE
  - mbus-protocol-aux.h, 30
- mbus\_read\_slave
  - mbus-protocol-aux.c, 24
  - mbus-protocol-aux.h, 33
- mbus\_record
  - mbus-protocol-aux.h, 31
- mbus\_record\_free

- mbus-protocol-aux.c, 25
- mbus-protocol-aux.h, 34
- mbus\_record\_new
  - mbus-protocol-aux.c, 25
  - mbus-protocol-aux.h, 34
- mbus\_rcv\_frame
  - mbus-protocol-aux.c, 25
  - mbus-protocol-aux.h, 34
- mbus\_scan\_2nd\_address\_range
  - mbus-protocol-aux.c, 25
  - mbus-protocol-aux.h, 34
- mbus\_send\_data\_request\_frame
  - mbus-protocol-aux.h, 34
- mbus\_send\_frame
  - mbus-protocol-aux.c, 26
  - mbus-protocol-aux.h, 35
- mbus\_send\_ping\_frame
  - mbus-protocol-aux.c, 26
- mbus\_send\_request\_frame
  - mbus-protocol-aux.c, 26
- mbus\_send\_select\_frame
  - mbus-protocol-aux.c, 26
  - mbus-protocol-aux.h, 35
- mbus\_serial\_connect
  - mbus-serial.c, 57
  - mbus-serial.h, 58
- mbus\_serial\_disconnect
  - mbus-serial.c, 57
  - mbus-serial.h, 58
- mbus\_serial\_handle
  - mbus-serial.h, 58
- mbus\_serial\_rcv\_frame
  - mbus-serial.c, 58
  - mbus-serial.h, 59
- mbus\_serial\_send\_frame
  - mbus-serial.c, 58
  - mbus-serial.h, 59
- mbus\_slave\_data
  - mbus-protocol.h, 52
- mbus\_slave\_data\_get
  - mbus-protocol.c, 44
  - mbus-protocol.h, 56
- mbus\_string
  - mbus-protocol-aux.h, 31
- mbus\_tcp\_connect
  - mbus-tcp.c, 59
  - mbus-tcp.h, 60
- mbus\_tcp\_disconnect
  - mbus-tcp.c, 59
  - mbus-tcp.h, 60
- mbus\_tcp\_handle
  - mbus-tcp.h, 60
- mbus\_tcp\_rcv\_frame
  - mbus-tcp.c, 60
- mbus-tcp.h, 61
- mbus\_tcp\_send\_frame
  - mbus-tcp.c, 60
  - mbus-tcp.h, 61
- mbus\_unit\_prefix
  - mbus-protocol.c, 44
  - mbus-protocol.h, 56
- mbus\_value
  - mbus-protocol-aux.h, 31
- mbus\_value\_information\_block
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_ADC
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_BUS
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_COLD\_-  
WATER
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
COMPR\_AIR
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_COOL\_IN
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_COOL\_-  
OUT
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_DUAL\_-  
WATER
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
ELECTRICITY
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_GAS
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_HEAT
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_HEAT\_-  
COST
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_HOT\_-  
WATER
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_OIL
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_OTHER
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_-  
PRESSURE
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_STEAM
  - mbus-protocol.h, 52
- MBUS\_VARIABLE\_DATA\_MEDIUM\_WATER
  - mbus-protocol.h, 52
- mbus\_variable\_value\_decode

- mbus-protocol-aux.c, 26
- mbus\_variable\_vif
  - mbus-protocol-aux.c, 23
- mbus\_vib\_unit\_lookup
  - mbus-protocol.c, 44
  - mbus-protocol.h, 56
- mbus\_vib\_unit\_normalize
  - mbus-protocol-aux.c, 26
  - mbus-protocol-aux.h, 35
- mbus\_vif\_unit\_lookup
  - mbus-protocol.c, 44
  - mbus-protocol.h, 56
- mbus\_vif\_unit\_normalize
  - mbus-protocol-aux.c, 27
  - mbus-protocol-aux.h, 35
- mdh
  - \_mbus\_data\_variable, 11
- medium
  - \_mbus\_data\_secondary\_address, 10
  - \_mbus\_data\_variable\_header, 12
- mfg\_data
  - \_mbus\_data\_variable, 11
- mfg\_data\_len
  - \_mbus\_data\_variable, 11
- ndife
  - \_mbus\_data\_information\_block, 9
- next
  - \_mbus\_data\_record, 9
- NITEMS
  - mbus-protocol.c, 39
  - mbus-protocol.h, 52
- nrecords
  - \_mbus\_data\_variable, 11
- nvife
  - \_mbus\_value\_information\_block, 19
- PACKET\_BUFF\_SIZE
  - mbus-serial.c, 57
  - mbus-tcp.c, 59
- parse\_debug
  - mbus-protocol.c, 44
- port
  - \_mbus\_tcp\_handle, 18
- primary
  - \_mbus\_address, 7
- quantity
  - \_mbus\_record, 15
  - \_mbus\_variable\_vif, 19
- real\_val
  - \_mbus\_value, 18
- record
  - \_mbus\_data\_variable, 11
- secondary
  - \_mbus\_address, 7
- signature
  - \_mbus\_data\_variable\_header, 12
- size
  - \_mbus\_string, 17
- slave\_data
  - mbus-protocol.c, 44
- sock
  - \_mbus\_tcp\_handle, 18
- start1
  - \_mbus\_frame, 13
- start2
  - \_mbus\_frame, 13
- state\_acd
  - \_mbus\_slave\_data, 16
- state\_fcb
  - \_mbus\_slave\_data, 16
- status
  - \_mbus\_data\_fixed, 8
  - \_mbus\_data\_variable\_header, 12
- stop
  - \_mbus\_frame, 13
- str\_val
  - \_mbus\_value, 18
- t
  - \_mbus\_serial\_handle, 16
- tx\_cnt
  - \_mbus\_data\_fixed, 8
- type
  - \_mbus\_frame, 13
  - \_mbus\_frame\_data, 13
- unit
  - \_mbus\_record, 15
  - \_mbus\_variable\_vif, 19
- value
  - \_mbus\_record, 15
  - \_mbus\_string, 17
- version
  - \_mbus\_data\_secondary\_address, 10
  - \_mbus\_data\_variable\_header, 12
- vib
  - \_mbus\_data\_record\_header, 10
- vif
  - \_mbus\_value\_information\_block, 19
  - \_mbus\_variable\_vif, 19
- vif\_table
  - mbus-protocol-aux.c, 27
- vife
  - \_mbus\_value\_information\_block, 19