

My Project

1.0

Generated by Doxygen 1.14.0

1 Data Structure Index	1
1.1 Data Structures	1
2 File Index	3
2.1 File List	3
3 Data Structure Documentation	5
3.1 bmcddata Struct Reference	5
3.1.1 Detailed Description	6
3.1.2 Field Documentation	6
3.1.2.1 adc_sample	6
3.1.2.2 b1_voltage	6
3.1.2.3 b2_voltage	6
3.1.2.4 battery_current	6
3.1.2.5 cc_current	6
3.1.2.6 cc_voltage	7
3.1.2.7 dac_sample	7
3.1.2.8 datain	7
3.1.2.9 dataout	7
3.1.2.10 input_voltage	7
3.1.2.11 logic_voltage	7
3.1.2.12 pv_current	7
3.1.2.13 pv_voltage	7
3.1.2.14 system_voltage	8
3.1.2.15 utc	8
3.2 didata Struct Reference	8
3.2.1 Detailed Description	8
3.2.2 Field Documentation	8
3.2.2.1 D0	8
3.2.2.2 D1	8
3.2.2.3 D2	9
3.2.2.4 D3	9
3.2.2.5 D4	9
3.2.2.6 D5	9
3.2.2.7 D6	9
3.2.2.8 D7	9
3.3 dio_buf_type Union Reference	10
3.3.1 Detailed Description	10
3.3.2 Field Documentation	10
3.3.2.1 d	10
3.3.2.2 dio_buf	10
3.4 energy_type Struct Reference	11
3.4.1 Detailed Description	11

3.4.2 Field Documentation	12
3.4.2.1 ac_low_adj	12
3.4.2.2 ac_mismatch	12
3.4.2.3 ac_sw_on	12
3.4.2.4 ac_sw_status	12
3.4.2.5 adc	12
3.4.2.6 client_ha	12
3.4.2.7 client_p	12
3.4.2.8 client_sd	13
3.4.2.9 comedi	13
3.4.2.10 dac	13
3.4.2.11 dc_mismatch	13
3.4.2.12 di_16b	13
3.4.2.13 dl_excess	13
3.4.2.14 dl_excess_adj	13
3.4.2.15 do_16b	13
3.4.2.16 dumpload	14
3.4.2.17 fm80	14
3.4.2.18 gti_delay	14
3.4.2.19 gti_low_adj	14
3.4.2.20 gti_sw_on	14
3.4.2.21 gti_sw_status	14
3.4.2.22 ha_lock	14
3.4.2.23 homeassistant	14
3.4.2.24 iammeter	15
3.4.2.25 im_delay	15
3.4.2.26 im_display	15
3.4.2.27 log_spam	15
3.4.2.28 log_time_reset	15
3.4.2.29 mode_mismatch	15
3.4.2.30 mqtt_count	15
3.4.2.31 once_ac	15
3.4.2.32 once_gti	16
3.4.2.33 once_gti_zero	16
3.4.2.34 rc	16
3.4.2.35 sane	16
3.4.2.36 sequence	16
3.4.2.37 solar_mode	16
3.4.2.38 solar_shutdown	16
3.4.2.39 speed_go	16
3.4.2.40 startup	17
3.4.2.41 thirty_sec_clock	17

3.5 ha_flag_type Struct Reference	17
3.5.1 Detailed Description	17
3.5.2 Field Documentation	17
3.5.2.1 deliveredtoken	17
3.5.2.2 energy_mode	17
3.5.2.3 ha_id	18
3.5.2.4 rec_ok	18
3.5.2.5 receivedtoken	18
3.5.2.6 runner	18
3.5.2.7 var_update	18
4 File Documentation	19
4.1 .dep.inc	19
4.2 bmc.c File Reference	19
4.2.1 Detailed Description	20
4.2.2 Function Documentation	20
4.2.2.1 led_lightshow()	20
4.2.2.2 main()	20
4.2.3 Variable Documentation	20
4.2.3.1 bmc	20
4.2.3.2 board_name	20
4.2.3.3 driver_name	21
4.2.3.4 E	21
4.2.3.5 fout	21
4.2.3.6 sine_wave	21
4.3 bmc.c	22
4.4 bmc.h	24
4.5 bmc_mqtt.c File Reference	25
4.5.1 Detailed Description	26
4.5.2 Macro Definition Documentation	26
4.5.2.1 COEF	26
4.5.3 Function Documentation	26
4.5.3.1 ac0_filter()	26
4.5.3.2 ac1_filter()	27
4.5.3.3 bmc_mqtt_init()	27
4.5.3.4 comedi_push_mqtt()	27
4.5.3.5 connlost()	27
4.5.3.6 delivered()	27
4.5.3.7 log_time()	27
4.5.3.8 mqtt_bmc_data()	28
4.5.3.9 msgarrvd()	28
4.5.3.10 showIP()	28

4.5.3.11 skeleton_daemon()	28
4.5.3.12 timer_callback()	28
4.5.4 Variable Documentation	28
4.5.4.1 conn_opts_ha	28
4.5.4.2 conn_opts_p	29
4.5.4.3 conn_opts_sd	29
4.5.4.4 ha_flag_vars_ss	29
4.5.4.5 hname	29
4.5.4.6 hname_len	29
4.5.4.7 hname_ptr	29
4.5.4.8 new_timer	30
4.5.4.9 old_timer	30
4.5.4.10 pubmsg	30
4.5.4.11 rawtime	30
4.5.4.12 token	30
4.6 bmc_mqtt.c	31
4.7 bmc_mqtt.h	36
4.8 bmc.o.d	37
4.9 bmc.o.d	37
4.10 bmc_mqtt.o.d	37
4.11 bmc_mqtt.o.d	37
4.12 daq.o.d	37
4.13 daq.o.d	37
4.14 daq.c	38
4.15 daq.h	42
4.16 mqtt_rec.h	43
4.17 mqtt_vars.h	44
4.18 c_standard_headers_indexer.c	44
4.19 cpp_standard_headers_indexer.cpp	45

Index	49
--------------	-----------

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

bmcddata	5
didata	8
dio_buf_type	10
energy_type	11
ha_flag_type	17

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

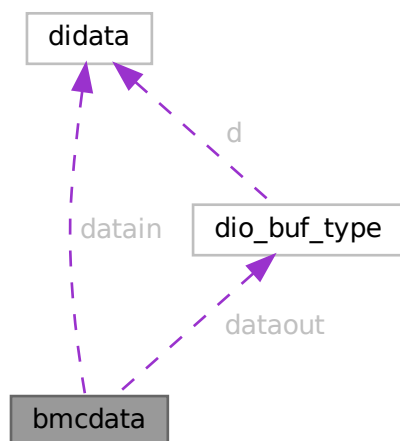
.dep.inc	19
bmc.c	19
bmc.h	24
bmc_mqtt.c	25
bmc_mqtt.h	36
daq.c	38
daq.h	42
mqtt_rec.h	43
mqtt_vars.h	44
build/Debug/GNU-Linux/bmc.o.d	37
build/Debug/GNU-Linux/bmc_mqtt.o.d	37
build/Debug/GNU-Linux/daq.o.d	37
build/Release/GNU-Linux/bmc.o.d	37
build/Release/GNU-Linux/bmc_mqtt.o.d	37
build/Release/GNU-Linux/daq.o.d	37
nbproject/private/c_standard_headers_indexer.c	44
nbproject/private/cpp_standard_headers_indexer.cpp	45

Chapter 3

Data Structure Documentation

3.1 bmcddata Struct Reference

Collaboration diagram for bmcddata:



Data Fields

- double [pv_voltage](#)
- double [cc_voltage](#)
- double [input_voltage](#)
- double [b1_voltage](#)
- double [b2_voltage](#)
- double [system_voltage](#)
- double [logic_voltage](#)
- double [pv_current](#)
- double [cc_current](#)

- double [battery_current](#)
- struct [didata](#) datain
- union [dio_buf_type](#) dataout
- int32_t [adc_sample](#) [32]
- int32_t [dac_sample](#) [32]
- int32_t [utc](#)

3.1.1 Detailed Description

Definition at line [52](#) of file [daq.h](#).

3.1.2 Field Documentation

3.1.2.1 [adc_sample](#)

```
int32_t adc_sample[32]
```

Definition at line [57](#) of file [daq.h](#).

3.1.2.2 [b1_voltage](#)

```
double b1_voltage
```

Definition at line [53](#) of file [daq.h](#).

3.1.2.3 [b2_voltage](#)

```
double b2_voltage
```

Definition at line [53](#) of file [daq.h](#).

3.1.2.4 [battery_current](#)

```
double battery_current
```

Definition at line [54](#) of file [daq.h](#).

3.1.2.5 [cc_current](#)

```
double cc_current
```

Definition at line [54](#) of file [daq.h](#).

3.1.2.6 cc_voltage

```
double cc_voltage
```

Definition at line 53 of file [daq.h](#).

3.1.2.7 dac_sample

```
int32_t dac_sample[32]
```

Definition at line 58 of file [daq.h](#).

3.1.2.8 datain

```
struct didata datain
```

Definition at line 55 of file [daq.h](#).

3.1.2.9 dataout

```
union dio_buf_type dataout
```

Definition at line 56 of file [daq.h](#).

3.1.2.10 input_voltage

```
double input_voltage
```

Definition at line 53 of file [daq.h](#).

3.1.2.11 logic_voltage

```
double logic_voltage
```

Definition at line 53 of file [daq.h](#).

3.1.2.12 pv_current

```
double pv_current
```

Definition at line 54 of file [daq.h](#).

3.1.2.13 pv_voltage

```
double pv_voltage
```

Definition at line 53 of file [daq.h](#).

3.1.2.14 system_voltage

```
double system_voltage
```

Definition at line 53 of file [daq.h](#).

3.1.2.15 utc

```
int32_t utc
```

Definition at line 59 of file [daq.h](#).

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

- [daq.h](#)

3.2 didata Struct Reference

Data Fields

- uint32_t [D0](#): 1
- uint32_t [D1](#): 1
- uint32_t [D2](#): 1
- uint32_t [D3](#): 1
- uint32_t [D4](#): 1
- uint32_t [D5](#): 1
- uint32_t [D6](#): 1
- uint32_t [D7](#): 1

3.2.1 Detailed Description

Definition at line 36 of file [daq.h](#).

3.2.2 Field Documentation

3.2.2.1 D0

```
uint32_t D0
```

Definition at line 37 of file [daq.h](#).

3.2.2.2 D1

```
uint32_t D1
```

Definition at line 38 of file [daq.h](#).

3.2.2.3 D2

`uint32_t D2`

Definition at line 39 of file [daq.h](#).

3.2.2.4 D3

`uint32_t D3`

Definition at line 40 of file [daq.h](#).

3.2.2.5 D4

`uint32_t D4`

Definition at line 41 of file [daq.h](#).

3.2.2.6 D5

`uint32_t D5`

Definition at line 42 of file [daq.h](#).

3.2.2.7 D6

`uint32_t D6`

Definition at line 43 of file [daq.h](#).

3.2.2.8 D7

`uint32_t D7`

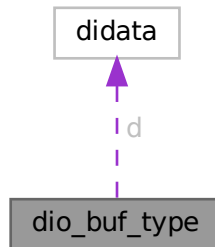
Definition at line 44 of file [daq.h](#).

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

- [daq.h](#)

3.3 dio_buf_type Union Reference

Collaboration diagram for dio_buf_type:



Data Fields

- uint32_t [dio_buf](#)
- struct [didata](#) [d](#)

3.3.1 Detailed Description

Definition at line 47 of file [daq.h](#).

3.3.2 Field Documentation

3.3.2.1 d

```
struct didata d
```

Definition at line 49 of file [daq.h](#).

3.3.2.2 dio_buf

```
uint32_t dio_buf
```

Definition at line 48 of file [daq.h](#).

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

- [daq.h](#)

3.4 energy_type Struct Reference

Data Fields

- volatile bool [once_gti](#)
- volatile bool [once_ac](#)
- volatile bool [iammeter](#)
- volatile bool [fm80](#)
- volatile bool [dumpload](#)
- volatile bool [homeassistant](#)
- volatile bool [once_gti_zero](#)
- volatile bool [comedi](#)
- volatile double [gti_low_adj](#)
- volatile double [ac_low_adj](#)
- volatile double [dl_excess_adj](#)
- volatile bool [ac_sw_on](#)
- volatile bool [gti_sw_on](#)
- volatile bool [ac_sw_status](#)
- volatile bool [gti_sw_status](#)
- volatile bool [solar_shutdown](#)
- volatile bool [solar_mode](#)
- volatile bool [startup](#)
- volatile bool [ac_mismatch](#)
- volatile bool [dc_mismatch](#)
- volatile bool [mode_mismatch](#)
- volatile bool [dl_excess](#)
- volatile uint32_t [speed_go](#)
- volatile uint32_t [im_delay](#)
- volatile uint32_t [im_display](#)
- volatile uint32_t [gti_delay](#)
- volatile uint32_t [sequence](#)
- volatile uint32_t [mqtt_count](#)
- volatile int32_t [rc](#)
- volatile int32_t [sane](#)
- volatile uint32_t [thirty_sec_clock](#)
- volatile uint32_t [log_spam](#)
- volatile uint32_t [log_time_reset](#)
- pthread_mutex_t [ha_lock](#)
- volatile int16_t [di_16b](#)
- volatile int16_t [do_16b](#)
- double [adc](#) [16]
- double [dac](#) [16]
- MQTTClient [client_p](#)
- MQTTClient [client_sd](#)
- MQTTClient [client_ha](#)

3.4.1 Detailed Description

Definition at line 90 of file [bmc.h](#).

3.4.2 Field Documentation

3.4.2.1 `ac_low_adj`

```
volatile double ac_low_adj
```

Definition at line 92 of file [bmc.h](#).

3.4.2.2 `ac_mismatch`

```
volatile bool ac_mismatch
```

Definition at line 93 of file [bmc.h](#).

3.4.2.3 `ac_sw_on`

```
volatile bool ac_sw_on
```

Definition at line 93 of file [bmc.h](#).

3.4.2.4 `ac_sw_status`

```
volatile bool ac_sw_status
```

Definition at line 93 of file [bmc.h](#).

3.4.2.5 `adc`

```
double adc[16]
```

Definition at line 99 of file [bmc.h](#).

3.4.2.6 `client_ha`

```
MQTTClient client_ha
```

Definition at line 100 of file [bmc.h](#).

3.4.2.7 `client_p`

```
MQTTClient client_p
```

Definition at line 100 of file [bmc.h](#).

3.4.2.8 client_sd

```
MQTTClient client_sd
```

Definition at line 100 of file [bmc.h](#).

3.4.2.9 comedi

```
volatile bool comedi
```

Definition at line 91 of file [bmc.h](#).

3.4.2.10 dac

```
double dac[16]
```

Definition at line 99 of file [bmc.h](#).

3.4.2.11 dc_mismatch

```
volatile bool dc_mismatch
```

Definition at line 93 of file [bmc.h](#).

3.4.2.12 di_16b

```
volatile int16_t di_16b
```

Definition at line 98 of file [bmc.h](#).

3.4.2.13 dl_excess

```
volatile bool dl_excess
```

Definition at line 93 of file [bmc.h](#).

3.4.2.14 dl_excess_adj

```
volatile double dl_excess_adj
```

Definition at line 92 of file [bmc.h](#).

3.4.2.15 do_16b

```
volatile int16_t do_16b
```

Definition at line 98 of file [bmc.h](#).

3.4.2.16 **dumpload**

```
volatile bool dumpload
```

Definition at line 91 of file [bmc.h](#).

3.4.2.17 **fm80**

```
volatile bool fm80
```

Definition at line 91 of file [bmc.h](#).

3.4.2.18 **gti_delay**

```
volatile uint32_t gti_delay
```

Definition at line 94 of file [bmc.h](#).

3.4.2.19 **gti_low_adj**

```
volatile double gti_low_adj
```

Definition at line 92 of file [bmc.h](#).

3.4.2.20 **gti_sw_on**

```
volatile bool gti_sw_on
```

Definition at line 93 of file [bmc.h](#).

3.4.2.21 **gti_sw_status**

```
volatile bool gti_sw_status
```

Definition at line 93 of file [bmc.h](#).

3.4.2.22 **ha_lock**

```
pthread_mutex_t ha_lock
```

Definition at line 97 of file [bmc.h](#).

3.4.2.23 **homeassistant**

```
volatile bool homeassistant
```

Definition at line 91 of file [bmc.h](#).

3.4.2.24 iammeter

```
volatile bool iammeter
```

Definition at line 91 of file [bmc.h](#).

3.4.2.25 im_delay

```
volatile uint32_t im_delay
```

Definition at line 94 of file [bmc.h](#).

3.4.2.26 im_display

```
volatile uint32_t im_display
```

Definition at line 94 of file [bmc.h](#).

3.4.2.27 log_spam

```
volatile uint32_t log_spam
```

Definition at line 96 of file [bmc.h](#).

3.4.2.28 log_time_reset

```
volatile uint32_t log_time_reset
```

Definition at line 96 of file [bmc.h](#).

3.4.2.29 mode_mismatch

```
volatile bool mode_mismatch
```

Definition at line 93 of file [bmc.h](#).

3.4.2.30 mqtt_count

```
volatile uint32_t mqtt_count
```

Definition at line 94 of file [bmc.h](#).

3.4.2.31 once_ac

```
volatile bool once_ac
```

Definition at line 91 of file [bmc.h](#).

3.4.2.32 once_gti

```
volatile bool once_gti
```

Definition at line 91 of file [bmc.h](#).

3.4.2.33 once_gti_zero

```
volatile bool once_gti_zero
```

Definition at line 91 of file [bmc.h](#).

3.4.2.34 rc

```
volatile int32_t rc
```

Definition at line 95 of file [bmc.h](#).

3.4.2.35 sane

```
volatile int32_t sane
```

Definition at line 95 of file [bmc.h](#).

3.4.2.36 sequence

```
volatile uint32_t sequence
```

Definition at line 94 of file [bmc.h](#).

3.4.2.37 solar_mode

```
volatile bool solar_mode
```

Definition at line 93 of file [bmc.h](#).

3.4.2.38 solar_shutdown

```
volatile bool solar_shutdown
```

Definition at line 93 of file [bmc.h](#).

3.4.2.39 speed_go

```
volatile uint32_t speed_go
```

Definition at line 94 of file [bmc.h](#).

3.4.2.40 startup

```
volatile bool startup
```

Definition at line 93 of file [bmc.h](#).

3.4.2.41 thirty_sec_clock

```
volatile uint32_t thirty_sec_clock
```

Definition at line 96 of file [bmc.h](#).

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

- [bmc.h](#)

3.5 ha_flag_type Struct Reference

Data Fields

- volatile MQTTClient_deliveryToken [deliveredtoken](#)
- volatile MQTTClient_deliveryToken [receivedtoken](#)
- volatile bool [runner](#)
- volatile bool [rec_ok](#)
- int32_t [ha_id](#)
- volatile int32_t [var_update](#)
- volatile int32_t [energy_mode](#)

3.5.1 Detailed Description

Definition at line 30 of file [mqtt_rec.h](#).

3.5.2 Field Documentation

3.5.2.1 deliveredtoken

```
volatile MQTTClient_deliveryToken deliveredtoken
```

Definition at line 31 of file [mqtt_rec.h](#).

3.5.2.2 energy_mode

```
volatile int32_t energy_mode
```

Definition at line 34 of file [mqtt_rec.h](#).

3.5.2.3 ha_id

```
int32_t ha_id
```

Definition at line 33 of file [mqtt_rec.h](#).

3.5.2.4 rec_ok

```
volatile bool rec_ok
```

Definition at line 32 of file [mqtt_rec.h](#).

3.5.2.5 receivedtoken

```
volatile MQTTClient_deliveryToken receivedtoken
```

Definition at line 31 of file [mqtt_rec.h](#).

3.5.2.6 runner

```
volatile bool runner
```

Definition at line 32 of file [mqtt_rec.h](#).

3.5.2.7 var_update

```
volatile int32_t var_update
```

Definition at line 34 of file [mqtt_rec.h](#).

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

- [mqtt_rec.h](#)

Chapter 4

File Documentation

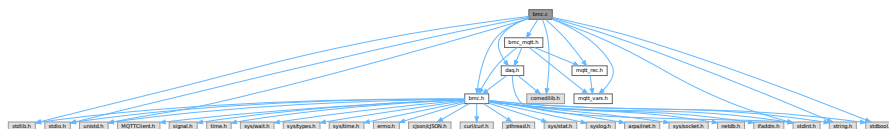
4.1 .dep.inc

```
00001 # This code depends on make tool being used
00002 DEFFILES=$(wildcard $(addsuffix .d, ${OBJECTFILES} ${TESTOBJECTFILES}))
00003 ifneq (${DEFFILES},)
00004 include ${DEFFILES}
00005 endif
```

4.2 bmc.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include <comedilib.h>
#include "daq.h"
#include "bmc.h"
#include "mqtt_rec.h"
#include "mqtt_vars.h"
#include "bmc_mqtt.h"
```

Include dependency graph for bmc.c:



Functions

- void [led_lightshow](#) (int speed)
- int [main](#) (int argc, char *argv[])

Variables

- volatile struct [bmcddata](#) [bmc](#)
- struct [energy_type](#) [E](#)
- const char * [board_name](#) = "NO_BOARD"
- const char * [driver_name](#) = "NO_DRIVER"
- FILE * [fout](#)
- uint8_t [sine_wave](#) [256]

4.2.1 Detailed Description

Demo code for driver testing, a simple console display of data inputs and voltage

This file may be freely modified, distributed, and combined with other software, as long as proper attribution is given in the source code.

Definition in file [bmc.c](#).

4.2.2 Function Documentation

4.2.2.1 [led_lightshow\(\)](#)

```
void led_lightshow (  
    int speed)
```

Definition at line [96](#) of file [bmc.c](#).

4.2.2.2 [main\(\)](#)

```
int main (  
    int argc,  
    char * argv[])
```

Definition at line [130](#) of file [bmc.c](#).

4.2.3 Variable Documentation

4.2.3.1 [bmc](#)

```
volatile struct bmcddata bmc
```

Definition at line [22](#) of file [bmc.c](#).

4.2.3.2 [board_name](#)

```
const char* board\_name = "NO_BOARD"
```

Definition at line [55](#) of file [bmc.c](#).

4.2.3.3 driver_name

```
const char * driver_name = "NO_DRIVER"
```

Definition at line 55 of file [bmc.c](#).

4.2.3.4 E

```
struct energy_type E
```

Initial value:

```
= {  
    .once_gti = true,  
    .once_ac = true,  
    .once_gti_zero = true,  
    .iammeter = false,  
    .fm80 = false,  
    .dumpload = false,  
    .homeassistant = false,  
    .ac_low_adj = 0.0f,  
    .gti_low_adj = 0.0f,  
    .ac_sw_on = true,  
    .gti_sw_on = true,  
    .im_delay = 0,  
    .gti_delay = 0,  
    .im_display = 0,  
    .rc = 0,  
    .speed_go = 0,  
    .ac_sw_status = false,  
    .gti_sw_status = false,  
    .solar_mode = false,  
    .solar_shutdown = false,  
    .startup = true,  
    .ac_mismatch = false,  
    .dc_mismatch = false,  
    .mode_mismatch = false,  
    .dl_excess = false,  
    .dl_excess_adj = 0.0f,  
}
```

Definition at line 24 of file [bmc.c](#).

4.2.3.5 fout

```
FILE* fout
```

Definition at line 57 of file [bmc.c](#).

4.2.3.6 sine_wave

```
uint8_t sine_wave[256]
```

Definition at line 61 of file [bmc.c](#).

4.3 bmc.c

[Go to the documentation of this file.](#)

```

00001
00008
00009 #include <stdlib.h>
00010 #include <stdio.h> /* for printf() */
00011 #include <unistd.h>
00012 #include <stdint.h>
00013 #include <string.h>
00014 #include <stdbool.h>
00015 #include <comedilib.h>
00016 #include "daq.h"
00017 #include "bmc.h"
00018 #include "mgtt_rec.h"
00019 #include "mgtt_vars.h"
00020 #include "bmc_mgtt.h"
00021
00022 volatile struct bmcddata bmc; /* DIO buffer */
00023
00024 struct energy_type E = {
00025     .once_gti = true,
00026     .once_ac = true,
00027     .once_gti_zero = true,
00028     .iammeter = false,
00029     .fm80 = false,
00030     .dumpload = false,
00031     .homeassistant = false,
00032     .ac_low_adj = 0.0f,
00033     .gti_low_adj = 0.0f,
00034     .ac_sw_on = true,
00035     .gti_sw_on = true,
00036     .im_delay = 0,
00037     .gti_delay = 0,
00038     .im_display = 0,
00039     .rc = 0,
00040     .speed_go = 0,
00041     .ac_sw_status = false,
00042     .gti_sw_status = false,
00043     .solar_mode = false,
00044     .solar_shutdown = false,
00045     .startup = true,
00046     .ac_mismatch = false,
00047     .dc_mismatch = false,
00048     .mode_mismatch = false,
00049     .dl_excess = false,
00050     .dl_excess_adj = 0.0f,
00051 };
00052
00053
00054 // Comedi I/O device type
00055 const char *board_name = "NO_BOARD", *driver_name = "NO_DRIVER";
00056
00057 FILE* fout; // logging stream
00058
00059 /* ripped from http://aquaticus.info/pwm-sine-wave */
00060
00061 uint8_t sine_wave[256] = {
00062     0x80, 0x83, 0x86, 0x89, 0x8C, 0x90, 0x93, 0x96,
00063     0x99, 0x9C, 0x9F, 0xA2, 0xA5, 0xA8, 0xAB, 0xAE,
00064     0xB1, 0xB3, 0xB6, 0xB9, 0xBC, 0xBF, 0xC1, 0xC4,
00065     0xC7, 0xC9, 0xCC, 0xCE, 0xD1, 0xD3, 0xD5, 0xD8,
00066     0xDA, 0xDC, 0xDE, 0xE0, 0xE2, 0xE4, 0xE6, 0xE8,
00067     0xEA, 0xEB, 0xED, 0xEF, 0xF0, 0xF1, 0xF3, 0xF4,
00068     0xF5, 0xF6, 0xF8, 0xF9, 0xFA, 0xFA, 0xFB, 0xFC,
00069     0xFD, 0xFD, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF,
00070     0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFE, 0xFD,
00071     0xFD, 0xFC, 0xFB, 0xFA, 0xFA, 0xF9, 0xF8, 0xF6,
00072     0xF5, 0xF4, 0xF3, 0xF1, 0xF0, 0xEF, 0xED, 0xEB,
00073     0xEA, 0xE8, 0xE6, 0xE4, 0xE2, 0xE0, 0xDE, 0xDC,
00074     0xDA, 0xD8, 0xD5, 0xD3, 0xD1, 0xCE, 0xCC, 0xC9,
00075     0xC7, 0xC4, 0xC1, 0xBF, 0xBC, 0xB9, 0xB6, 0xB3,
00076     0xB1, 0xAE, 0xAB, 0xA8, 0xA5, 0xA2, 0x9F, 0x9C,
00077     0x99, 0x96, 0x93, 0x90, 0x8C, 0x89, 0x86, 0x83,
00078     0x80, 0x7D, 0x7A, 0x77, 0x74, 0x70, 0x6D, 0x6A,
00079     0x67, 0x64, 0x61, 0x5E, 0x5B, 0x58, 0x55, 0x52,
00080     0x4F, 0x4D, 0x4A, 0x47, 0x44, 0x41, 0x3F, 0x3C,
00081     0x39, 0x37, 0x34, 0x32, 0x2F, 0x2D, 0x2B, 0x28,
00082     0x26, 0x24, 0x22, 0x20, 0x1E, 0x1C, 0x1A, 0x18,
00083     0x16, 0x15, 0x13, 0x11, 0x10, 0x0F, 0x0D, 0x0C,
00084     0x0B, 0x0A, 0x08, 0x07, 0x06, 0x06, 0x05, 0x04,
00085     0x03, 0x03, 0x02, 0x02, 0x02, 0x01, 0x01, 0x01,
00086     0x01, 0x01, 0x01, 0x01, 0x02, 0x02, 0x02, 0x03,
00087     0x03, 0x04, 0x05, 0x06, 0x06, 0x07, 0x08, 0x0A,
00088     0x0B, 0x0C, 0x0D, 0x0F, 0x10, 0x11, 0x13, 0x15,

```

```

00089     0x16, 0x18, 0x1A, 0x1C, 0x1E, 0x20, 0x22, 0x24,
00090     0x26, 0x28, 0x2B, 0x2D, 0x2F, 0x32, 0x34, 0x37,
00091     0x39, 0x3C, 0x3F, 0x41, 0x44, 0x47, 0x4A, 0x4D,
00092     0x4F, 0x52, 0x55, 0x58, 0x5B, 0x5E, 0x61, 0x64,
00093     0x67, 0x6A, 0x6D, 0x70, 0x74, 0x77, 0x7A, 0x7D
00094 };
00095
00096 void led_lightshow(int speed)
00097 {
00098     static int j = 0;
00099     static uint8_t cylon = 0xff;
00100     static int alive_led = 0;
00101     static bool LED_UP = true;
00102
00103     if (j++ >= speed) { // delay a bit ok
00104         if (0) { // screen status feedback
00105             bmc.dataout.dio_buf = ~cylon; // roll leds cylon style
00106         } else {
00107             bmc.dataout.dio_buf = cylon; // roll leds cylon style (inverted)
00108         }
00109
00110         if (LED_UP && (alive_led != 0)) {
00111             alive_led = alive_led * 2;
00112             cylon = cylon << 1;
00113         } else {
00114             if (alive_led != 0) alive_led = alive_led / 2;
00115             cylon = cylon >> 1;
00116         }
00117         if (alive_led < 2) {
00118             alive_led = 2;
00119             LED_UP = true;
00120         } else {
00121             if (alive_led > 128) {
00122                 alive_led = 128;
00123                 LED_UP = false;
00124             }
00125         }
00126         j = 0;
00127     }
00128 }
00129
00130 int main(int argc, char *argv[])
00131 {
00132     int do_ao_only = false;
00133     uint8_t i = 0, j = 75;
00134
00135     /*
00136      * start the MQTT processing
00137      */
00138     bmc_mqtt_init();
00139
00140     if (do_ao_only) {
00141         if (init_dac(0.0, 25.0, false) < 0) {
00142             fprintf(fout, "Missing Analog AO subdevice\n");
00143             return -1;
00144         }
00145
00146         while (true) {
00147             set_dac_raw(0, sine_wave[255 - i++] << 4);
00148             set_dac_raw(1, sine_wave[255 - j++] << 4);
00149         }
00150     } else {
00151 #ifndef DIGITAL_ONLY
00152         if (init_daq(0.0, 25.0, false) < 0) {
00153             fprintf(fout, "Missing Analog subdevice(s)\n");
00154             return -1;
00155         }
00156 #endif
00157         if (init_dio() < 0) {
00158             fprintf(fout, "Missing Digital subdevice(s)\n");
00159             return -1;
00160         }
00161
00162         E.dac[0] = 1.23f;
00163         E.dac[1] = 3.21f;
00164
00165         E.do_16b = 0x01;
00166         E.di_16b = 0x10;
00167
00168         fflush(fout);
00169         while (true) {
00170             usleep(MAIN_DELAY); // sample rate ~1 msec
00171             get_data_sample();
00172             if (!bmc.datain.D0) {
00173                 led_lightshow(1);
00174             }
00175         }
00176     }

```

```

00176         }
00177         if (ha_flag_vars_ss.runner) { // 30 second timer or trigger from mqtt
00178             comedi_push_mqtt(); // send json formatted data to the mqtt server
00179             ha_flag_vars_ss.runner = false;
00180         }
00181     }
00182 }
00183 }
00184     return 0;
00185 }
00186
00187

```

4.4 bmc.h

```

00001 /*
00002  * File:   bmc.h
00003  * Author: root
00004  *
00005  * Created on September 21, 2012, 12:54 PM
00006  */
00007
00008 #ifndef BMC_H
00009 #define BMC_H
00010
00011 #ifdef __cplusplus
00012 extern "C" {
00013 #endif
00014
00015 #include <stdlib.h>
00016 #include <stdio.h> /* for printf() */
00017 #include <unistd.h>
00018 #include <stdint.h>
00019 #include <string.h>
00020 #include <stdbool.h>
00021 #include <signal.h>
00022 #include <time.h>
00023 #include <sys/wait.h>
00024 #include <sys/types.h>
00025 #include <sys/time.h>
00026 #include <errno.h>
00027 #include <cjson/cJSON.h>
00028 #include <curl/curl.h>
00029 #include <pthread.h>
00030 #include <sys/stat.h>
00031 #include <syslog.h>
00032 #include <arpa/inet.h>
00033 #include <sys/socket.h>
00034 #include <netdb.h>
00035 #include <ifaddrs.h>
00036 #include "MQTTClient.h"
00037
00038 #define LOG_VERSION "V0.03"
00039 #define MQTT_VERSION "V3.11"
00040 #define TNAME "maint9"
00041 #define LADDRESS "tcp://127.0.0.1:1883"
00042 #ifdef __amd64
00043 #define ADDRESS "tcp://10.1.1.172:1883"
00044 #else
00045 #define ADDRESS "tcp://10.1.1.172:1883"
00046 #endif
00047 #define CLIENTID1 "Energy_Mqtt_BMC1"
00048 #define CLIENTID2 "Energy_Mqtt_BMC2"
00049 #define CLIENTID3 "Energy_Mqtt_BMC3"
00050 #define TOPIC_P "comedi/bmc/data/bmc"
00051 #define TOPIC_SPAM "comedi/bmc/data/spam"
00052 #define TOPIC_PACA "home-assistant/comedi/bmc"
00053 // #define TOPIC_PACB "mateq84/data/#"
00054 #define TOPIC_AI "comedi/bmc/data/ai"
00055 #define TOPIC_AO "comedi/bmc/data/ao"
00056 #define TOPIC_DI "comedi/bmc/data/di"
00057 #define TOPIC_DO "comedi/bmc/data/do"
00058 #define QOS 1
00059
00060 // #define TOPIC_SS "mateq84/data/solar" // receive data testing
00061
00062 #define TIMEOUT 10000L
00063 #define SPACING_USEC 500 * 1000
00064 #define USEC_SEC 1000000L
00065
00066 #define CMD_SEC 1
00067 #define TIME_SYNC_SEC 30
00068

```


- char * [log_time](#) (bool log)
- void [timer_callback](#) (int32_t signum)
- void [connlost](#) (void *context, char *cause)
- void [delivered](#) (void *context, MQTTClient_deliveryToken dt)
- void [bmc_mqtt_init](#) (void)
- int32_t [msgarrvd](#) (void *context, char *topicName, int topicLen, MQTTClient_message *message)
- void [mqtt_bmc_data](#) (MQTTClient client_p, const char *topic_p)
- void [comedi_push_mqtt](#) (void)

Variables

- struct itimerval [new_timer](#)
- struct itimerval [old_timer](#)
- time_t [rawtime](#)
- MQTTClient_connectOptions [conn_opts_p](#) = MQTTClient_connectOptions_initializer
- MQTTClient_connectOptions [conn_opts_sd](#) = MQTTClient_connectOptions_initializer
- MQTTClient_connectOptions [conn_opts_ha](#) = MQTTClient_connectOptions_initializer
- MQTTClient_message [pubmsg](#) = MQTTClient_message_initializer
- MQTTClient_deliveryToken [token](#)
- char [hname](#) [256]
- char * [hname_ptr](#) = hname
- size_t [hname_len](#) = 12
- struct [ha_flag_type](#) [ha_flag_vars_ss](#)

4.5.1 Detailed Description

show all assigned networking addresses and types on the current machine

Definition in file [bmc_mqtt.c](#).

4.5.2 Macro Definition Documentation

4.5.2.1 COEF

```
#define COEF 12.0f
```

Definition at line 3 of file [bmc_mqtt.c](#).

4.5.3 Function Documentation

4.5.3.1 ac0_filter()

```
double ac0_filter (
    const double raw)
```

Definition at line 461 of file [bmc_mqtt.c](#).

4.5.3.2 ac1_filter()

```
double ac1_filter (  
    const double raw)
```

Definition at line 470 of file [bmc_mqtt.c](#).

4.5.3.3 bmc_mqtt_init()

```
void bmc_mqtt_init (  
    void )
```

Definition at line 221 of file [bmc_mqtt.c](#).

4.5.3.4 comedi_push_mqtt()

```
void comedi_push_mqtt (  
    void )
```

Definition at line 456 of file [bmc_mqtt.c](#).

4.5.3.5 connlost()

```
void connlost (  
    void * context,  
    char * cause)
```

trouble in River-city

Definition at line 169 of file [bmc_mqtt.c](#).

4.5.3.6 delivered()

```
void delivered (  
    void * context,  
    MQTTClient_deliveryToken dt)
```

Definition at line 207 of file [bmc_mqtt.c](#).

4.5.3.7 log_time()

```
char * log_time (  
    bool log)
```

Definition at line 133 of file [bmc_mqtt.c](#).

4.5.3.8 mqtt_bmc_data()

```
void mqtt_bmc_data (
    MQTTClient client_p,
    const char * topic_p)
```

Definition at line 359 of file [bmc_mqtt.c](#).

4.5.3.9 msgarrvd()

```
int32_t msgarrvd (
    void * context,
    char * topicName,
    int topicLen,
    MQTTClient_message * message)
```

Definition at line 282 of file [bmc_mqtt.c](#).

4.5.3.10 showIP()

```
void showIP (
    void )
```

Definition at line 40 of file [bmc_mqtt.c](#).

4.5.3.11 skeleton_daemon()

```
void skeleton_daemon (
    void )
```

Definition at line 74 of file [bmc_mqtt.c](#).

4.5.3.12 timer_callback()

```
void timer_callback (
    int32_t signum)
```

Definition at line 157 of file [bmc_mqtt.c](#).

4.5.4 Variable Documentation

4.5.4.1 conn_opts_ha

```
MQTTClient_connectOptions conn_opts_ha = MQTTClient_connectOptions_initializer
```

Definition at line 18 of file [bmc_mqtt.c](#).

4.5.4.2 conn_opts_p

```
MQTTClient_connectOptions conn_opts_p = MQTTClient_connectOptions_initializer
```

Definition at line 16 of file [bmc_mqtt.c](#).

4.5.4.3 conn_opts_sd

```
MQTTClient_connectOptions conn_opts_sd = MQTTClient_connectOptions_initializer
```

Definition at line 17 of file [bmc_mqtt.c](#).

4.5.4.4 ha_flag_vars_ss

```
struct ha_flag_type ha_flag_vars_ss
```

Initial value:

```
= {  
    .runner = false,  
    .receivedtoken = false,  
    .deliveredtoken = false,  
    .rec_ok = false,  
    .ha_id = COMEDI_ID,  
    .var_update = 0,  
}
```

Definition at line 24 of file [bmc_mqtt.c](#).

4.5.4.5 hname

```
char hname[256]
```

Definition at line 21 of file [bmc_mqtt.c](#).

4.5.4.6 hname_len

```
size_t hname_len = 12
```

Definition at line 22 of file [bmc_mqtt.c](#).

4.5.4.7 hname_ptr

```
char * hname_ptr = hname
```

Definition at line 21 of file [bmc_mqtt.c](#).

4.5.4.8 new_timer

```
struct itimerval new_timer
```

Initial value:

```
= {  
    .it_value.tv_sec = CMD_SEC,  
    .it_value.tv_usec = 0,  
    .it_interval.tv_sec = CMD_SEC,  
    .it_interval.tv_usec = 0,  
}
```

Definition at line 8 of file [bmc_mqtt.c](#).

4.5.4.9 old_timer

```
struct itimerval old_timer
```

Definition at line 14 of file [bmc_mqtt.c](#).

4.5.4.10 pubmsg

```
MQTTClient_message pubmsg = MQTTClient_message_initializer
```

Definition at line 19 of file [bmc_mqtt.c](#).

4.5.4.11 rawtime

```
time_t rawtime
```

Definition at line 15 of file [bmc_mqtt.c](#).

4.5.4.12 token

```
MQTTClient_deliveryToken token
```

Definition at line 20 of file [bmc_mqtt.c](#).

4.6 bmc_mqtt.c

[Go to the documentation of this file.](#)

```

00001 #include "bmc_mqtt.h"
00002
00003 #define COEF          12.0f
00004
00005 static const char *const FW_Date = __DATE__;
00006 static const char *const FW_Time = __TIME__;
00007
00008 struct itimerval new_timer = {
00009     .it_value.tv_sec = CMD_SEC,
00010     .it_value.tv_usec = 0,
00011     .it_interval.tv_sec = CMD_SEC,
00012     .it_interval.tv_usec = 0,
00013 };
00014 struct itimerval old_timer;
00015 time_t rawtime;
00016 MQTTClient_connectOptions conn_opts_p = MQTTClient_connectOptions_initializer,
00017     conn_opts_sd = MQTTClient_connectOptions_initializer,
00018     conn_opts_ha = MQTTClient_connectOptions_initializer;
00019 MQTTClient_message pubmsg = MQTTClient_message_initializer;
00020 MQTTClient_deliveryToken token;
00021 char hname[256], *hname_ptr = hname;
00022 size_t hname_len = 12;
00023
00024 struct ha_flag_type ha_flag_vars_ss = {
00025     .runner = false,
00026     .receivedtoken = false,
00027     .deliveredtoken = false,
00028     .rec_ok = false,
00029     .ha_id = COMEDI_ID,
00030     .var_update = 0,
00031 };
00032
00033 double ac0_filter(const double);
00034 double ac1_filter(const double);
00035
00040 void showIP(void)
00041 {
00042     struct ifaddrs *ifaddr, *ifa;
00043     int s;
00044     char host[NI_MAXHOST];
00045
00046     if (getifaddrs(&ifaddr) == -1) {
00047         perror("getifaddrs");
00048         exit(EXIT_FAILURE);
00049     }
00050
00051     for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
00052         if (ifa->ifa_addr == NULL)
00053             continue;
00054
00055         s = getnameinfo(ifa->ifa_addr, sizeof(struct sockaddr_in), host, NI_MAXHOST, NULL, 0,
00056             NI_NUMERICHOST);
00057
00058         if (ifa->ifa_addr->sa_family == AF_INET) {
00059             if (s != 0) {
00060                 exit(EXIT_FAILURE);
00061             }
00062             printf("\tInterface : <%s>\n", ifa->ifa_name);
00063             printf("\tAddress : <%s>\n", host);
00064         }
00065     }
00066     freeifaddrs(ifaddr);
00067 }
00068
00069 /*
00071  * setup ha_energy program to run as a background daemon
00072  * disconnect and exit foreground startup process
00073  */
00074 void skeleton_daemon(void)
00075 {
00076     pid_t pid;
00077
00078     /* Fork off the parent process */
00079     pid = fork();
00080
00081     /* An error occurred */
00082     if (pid < 0) {
00083         printf("\r\n%s DAEMON failure LOG Version %s : MQTT Version %s\r\n", log_time(false),
00084             LOG_VERSION, MQTT_VERSION);
00085         exit(EXIT_FAILURE);
00086     }

```

```

00085     }
00086
00087     /* Success: Let the parent terminate */
00088     if (pid > 0) {
00089         exit(EXIT_SUCCESS);
00090     }
00091
00092     /* On success: The child process becomes session leader */
00093     if (setsid() < 0) {
00094         exit(EXIT_FAILURE);
00095     }
00096
00097     /* Catch, ignore and handle signals */
00098     /*TODO: Implement a working signal handler */
00099     // signal(SIGCHLD, SIG_IGN);
00100     // signal(SIGHUP, SIG_IGN);
00101
00102     /* Fork off for the second time*/
00103     pid = fork();
00104
00105     /* An error occurred */
00106     if (pid < 0) {
00107         exit(EXIT_FAILURE);
00108     }
00109
00110     /* Success: Let the parent terminate */
00111     if (pid > 0) {
00112         exit(EXIT_SUCCESS);
00113     }
00114
00115     /* Set new file permissions */
00116     umask(0);
00117
00118     /* Change the working directory to the root directory */
00119     /* or another appropriated directory */
00120     chdir("/");
00121
00122     /* Close all open file descriptors */
00123     int x;
00124     for (x = sysconf(_SC_OPEN_MAX); x >= 0; x--) {
00125         close(x);
00126     }
00127 }
00128 }
00129
00130 /*
00131  * sent the current UTC to the Dump Load controller
00132  */
00133 char * log_time(bool log)
00134 {
00135     static char time_log[RBUF_SIZ] = {0};
00136     time_t rawtime_log;
00137     int32_t len = 0;
00138
00139     tzset();
00140     timezone = 0;
00141     daylight = 0;
00142     time(&rawtime_log);
00143     sprintf(time_log, "%s", ctime(&rawtime_log));
00144     len = strlen(time_log);
00145     time_log[len - 1] = 0; // munge out the return character
00146     if (log) {
00147         fprintf(fout, "%s ", time_log);
00148         fflush(fout);
00149     }
00150     return time_log;
00151 }
00152
00153 /*
00154  * data update timer flag
00155  * and CMD_SEC seconds software time clock
00156  */
00157 void timer_callback(int32_t signum)
00158 {
00159     signal(signum, timer_callback);
00160     ha_flag_vars_ss.runner = true;
00161     E.thirty_sec_clock++;
00162     E.log_time_reset++;
00163 }
00164 }
00165
00166 /*
00167  * MQTT Broker connection errors can be fatal
00168  */
00169 void connlost(void *context, char *cause)
00170 {
00171     struct ha_flag_type *ha_flag = context;

```

```

00172     int32_t id_num = ha_flag->ha_id;
00173     static uint32_t times = 0;
00174     char * where = "Missing Topic";
00175     char * what = "Reconnection Error";
00176
00177     // bug-out if no context variables passed to callback
00178     if (context == NULL) {
00179         id_num = -1;
00180         goto bugout;
00181     }
00182
00183     if (times++ > MQTT_RECONN) {
00184         goto bugout;
00185     } else {
00186         if (times > 1) {
00187             fprintf(fout, "%s Connection lost, retrying %d \n", log_time(false), times);
00188             fprintf(fout, "%s      cause: %s, h_id %d, c_id %d, %s \n", log_time(false), cause, id_num,
0, what);
00189             fprintf(fout, "%s MQTT DAEMON reconnect failure, LOG Version %s : MQTT Version %s\n",
log_time(false), LOG_VERSION, MQTT_VERSION);
00190         }
00191         fflush(fout);
00192         times = 0;
00193         return;
00194     }
00195
00196 bugout:
00197     fprintf(fout, "%s Connection lost, exit ha_energy program\n", log_time(false));
00198     fprintf(fout, "%s      cause: %s, h_id %d, c_id %d, %s \n", log_time(false), cause, id_num, 0,
where);
00199     fprintf(fout, "%s MQTT DAEMON context is NULL failure, LOG Version %s : MQTT Version %s\n",
log_time(false), LOG_VERSION, MQTT_VERSION);
00200     fflush(fout);
00201     exit(EXIT_FAILURE);
00202 }
00203
00204 /*
00205  * set the broker has message token
00206  */
00207 void delivered(void *context, MQTTClient_deliveryToken dt)
00208 {
00209     struct ha_flag_type *ha_flag = context;
00210
00211     // bug-out if no context variables passed to callback
00212     if (context == NULL) {
00213         return;
00214     }
00215     ha_flag->deliveredtoken = dt;
00216 }
00217
00221 void bmc_mqtt_init(void)
00222 {
00223     E.mqtt_count = 0;
00224     gethostname(hname, hname_len);
00225     hname[12] = 0;
00226     printf("\r\n LOG Version %s : MQTT Version %s : Host Name %s\r\n", LOG_VERSION, MQTT_VERSION,
hname);
00227     showIP();
00228     skeleton_daemon();
00229 #ifdef LOG_TO_FILE
00230     fout = fopen(LOG_TO_FILE, "a");
00231     if (fout == NULL) {
00232         fout = fopen(LOG_TO_FILE_ALT, "a");
00233         if (fout == NULL) {
00234             fout = stdout;
00235             printf("\r\n%s Unable to open LOG file %s \r\n", log_time(false), LOG_TO_FILE_ALT);
00236         }
00237     }
00238 #else
00239     fout = stdout;
00240 #endif
00241
00242     /*
00243      * set the timer for MQTT publishing sample speed
00244      * CMD_SEC sets the time in seconds
00245      */
00246     setitimer(ITIMER_REAL, &new_timer, &old_timer);
00247     signal(SIGALRM, timer_callback);
00248
00249     if (strncmp(hname, TNAME, 6) == 0) {
00250         MQTTClient_create(&E.client_p, LADDRESS, CLIENTID1,
MQTTCLIENT_PERSISTENCE_NONE, NULL);
00251         conn_opts_p.keepAliveInterval = KAI;
00252         conn_opts_p.cleansession = 1;
00253         hname_ptr = LADDRESS;
00254     } else {
00255         MQTTClient_create(&E.client_p, ADDRESS, CLIENTID1,

```

```

00257         MQTTCLIENT_PERSISTENCE_NONE, NULL);
00258         conn_opts_p.keepAliveInterval = KAI;
00259         conn_opts_p.cleansession = 1;
00260         hname_ptr = ADDRESS;
00261     }
00262
00263     fprintf(fout, "\r\n%s Connect MQTT server %s, %s\n", log_time(false), hname_ptr, CLIENTID1);
00264     fflush(fout);
00265     MQTTClient_setCallbacks(E.client_p, &ha_flag_vars_ss, connlost, msgarrvd, delivered);
00266     if ((E.rc = MQTTClient_connect(E.client_p, &conn_opts_p)) != MQTTCLIENT_SUCCESS) {
00267         fprintf(fout, "%s Failed to connect MQTT server, return code %d %s, %s\n", log_time(false),
00268             E.rc, hname_ptr, CLIENTID1);
00269         fflush(fout);
00270         pthread_mutex_destroy(&E.ha_lock);
00271         exit(EXIT_FAILURE);
00272     }
00273     MQTTClient_subscribe(E.client_p, TOPIC_P, QOS); // sub for testing data from the HA_Energy system
00274
00275     pubmsg.payload = "online";
00276     pubmsg.payloadlen = strlen("online");
00277     pubmsg.qos = QOS;
00278     pubmsg.retained = 0;
00279     ha_flag_vars_ss.deliveredtoken = 0;
00280 }
00281
00282 int32_t msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
00283 {
00284     int32_t i, ret = 1;
00285     const char* payloadptr;
00286     char buffer[MBMQTT];
00287     struct ha_flag_type *ha_flag = context;
00288
00289     E.mqtt_count++;
00290     // bug-out if no context variables passed to callback
00291     if (context == NULL) {
00292         ret = -1;
00293         goto null_exit;
00294     }
00295
00296 #ifdef DEBUG_REC
00297     fprintf(fout, "Message arrived\n");
00298 #endif
00299     /*
00300      * move the received message into a processing holding buffer
00301      */
00302     payloadptr = message->payload;
00303     for (i = 0; i < message->payloadlen; i++) {
00304         buffer[i] = *payloadptr++;
00305     }
00306     buffer[i] = 0; // make a null terminated C string
00307
00308     // parse the JSON data in the holding buffer
00309     cJSON *json = cJSON_ParseWithLength(buffer, message->payloadlen);
00310     if (json == NULL) {
00311         const char *error_ptr = cJSON_GetErrorPtr();
00312         if (error_ptr != NULL) {
00313             fprintf(fout, "%s Error: %s NULL cJSON pointer\n", log_time(false), error_ptr);
00314         }
00315         ret = -1;
00316         ha_flag->rec_ok = false;
00317         E.comedi = false;
00318         goto error_exit;
00319     }
00320
00321     /*
00322      * MQTT messages for COMEDI
00323      */
00324 #ifdef DEBUG_REC
00325     fprintf(fout, "COMEDI MQTT data\r\n");
00326 #endif
00327     cJSON *data_result = json;
00328
00329     data_result = cJSON_GetObjectItemCaseSensitive(json, "Comedi_Request");
00330
00331     if (cJSON_IsString(data_result) && (data_result->valuestring != NULL)) {
00332         fprintf(fout, "%s Comedi Trigger from MQTT server, Topic %s %s\n", log_time(false), topicName,
00333             data_result->valuestring);
00334         fflush(fout);
00335         ret = true;
00336     }
00337     E.comedi = true;
00338
00339     // done with processing MQTT async message, set state flags
00340     ha_flag->receivedtoken = true;
00341     ha_flag->rec_ok = true;
00342     ha_flag_vars_ss.runner = true; // send data in response to received message of any type

```



```

00342     /*
00343     * exit and delete/free resources. In steps depending of possible error conditions
00344     */
00345 error_exit:
00346     // delete the JSON object
00347     cJSON_Delete(json);
00348 null_exit:
00349     // free the MQTT objects
00350     MQTTClient_freeMessage(&message);
00351     MQTTClient_free(topicName);
00352     fflush(fout);
00353     return ret;
00354 }
00355
00356 /*
00357 * send Comedi variables MQTT host
00358 */
00359 void mqtt_bmc_data(MQTTClient client_p, const char * topic_p)
00360 {
00361     cJSON *json;
00362     time_t rawtime;
00363     static uint32_t spam = 0;
00364
00365     MQTTClient_message pubmsg = MQTTClient_message_initializer;
00366     MQTTClient_deliveryToken token;
00367     ha_flag_vars_ss.deliveredtoken = 0;
00368
00369     fprintf(fout, "%s Sending Comedi data to MQTT server, Topic %s\n", log_time(false), topic_p);
00370     fflush(fout);
00371
00372 #ifndef DIGITAL_ONLY
00373     E.adc[0] = ac0_filter(get_adc_volts(0)); // over-sample avg
00374     E.adc[0] = ac0_filter(get_adc_volts(0));
00375     E.adc[0] = ac0_filter(get_adc_volts(0));
00376     E.adc[0] = ac0_filter(get_adc_volts(0));
00377     E.adc[0] = ac0_filter(get_adc_volts(0));
00378     E.adc[0] = ac0_filter(get_adc_volts(0));
00379     E.adc[0] = ac0_filter(get_adc_volts(0));
00380     E.adc[0] = ac0_filter(get_adc_volts(0));
00381     E.adc[1] = ac1_filter(get_adc_volts(1));
00382     E.adc[1] = ac1_filter(get_adc_volts(1));
00383     E.adc[1] = ac1_filter(get_adc_volts(1));
00384     E.adc[1] = ac1_filter(get_adc_volts(1));
00385     E.adc[1] = ac1_filter(get_adc_volts(1));
00386     E.adc[1] = ac1_filter(get_adc_volts(1));
00387     E.adc[1] = ac1_filter(get_adc_volts(1));
00388     E.adc[1] = ac1_filter(get_adc_volts(1));
00389
00390 #ifdef DAC_TESTING
00391     E.dac[0] = E.adc[0];
00392     E.dac[1] = E.adc[1];
00393 #endif
00394
00395 #ifndef DAC_TESTING
00396     set_dac_raw(0, 0);
00397     set_dac_raw(1, 0);
00398 #else
00399     set_dac_volts(0, E.adc[0]);
00400     set_dac_volts(1, E.adc[1]);
00401 #endif
00402 #endif
00403
00404     E.do_16b = bmc.dataout.dio_buf;
00405     E.di_16b = get_dio_bit(0) + (get_dio_bit(1) << 1) + (get_dio_bit(2) << 2) + (get_dio_bit(3) << 3) +
00406         (get_dio_bit(4) << 4);
00407
00408     E.mqtt_count++;
00409     E.sequence++;
00410     json = cJSON_CreateObject();
00411     cJSON_AddStringToObject(json, "RDAQ1name", CLIENTID1);
00412     cJSON_AddNumberToObject(json, "RDAQ1sequence", E.sequence);
00413     cJSON_AddNumberToObject(json, "RDAQ1mqtt_do_16b", (double) E.do_16b);
00414     cJSON_AddNumberToObject(json, "RDAQ1http_di_16b", (double) E.di_16b);
00415     cJSON_AddNumberToObject(json, "RDAQ1bmc_adc0", E.adc[0]);
00416     cJSON_AddNumberToObject(json, "RDAQ1bmc_adc1", E.adc[1]);
00417     cJSON_AddNumberToObject(json, "RDAQ1bmc_dac0", E.dac[0]);
00418     cJSON_AddNumberToObject(json, "RDAQ1bmc_dac1", E.dac[1]);
00419     cJSON_AddStringToObject(json, "RDAQ1build_date", FW_Date);
00420     cJSON_AddStringToObject(json, "RDAQ1build_time", FW_Time);
00421     time(&rawtime);
00422     cJSON_AddNumberToObject(json, "RDAQ1sequence_time", (double) rawtime);
00423     // convert the cJSON object to a JSON string
00424     char *json_str = cJSON_Print(json);
00425
00426     pubmsg.payload = json_str;
00427     pubmsg.payloadlen = strlen(json_str);
00428     pubmsg.qos = QOS;

```

```

00428     pubmsg.retained = 0;
00429
00430     MQTTClient_publishMessage(client_p, topic_p, &pubmsg, &token);
00431     // a busy, wait loop for the async delivery thread to complete
00432     {
00433         uint32_t waiting = 0;
00434         while (ha_flag_vars_ss.deliveredtoken != token) {
00435             usleep(TOKEN_DELAY);
00436             if (waiting++ > MQTT_RETRY) {
00437                 if (spam++ > 1) {
00438                     fprintf(fout, "%s SW mqtt_bmc_data, Still Waiting, timeout\r\n", log_time(false));
00439                     fflush(fout);
00440                     spam = 0;
00441                 }
00442                 break;
00443             } else {
00444                 spam = 0;
00445             }
00446         };
00447     }
00448
00449     cJSON_free(json_str);
00450     cJSON_Delete(json);
00451 }
00452
00453 /*
00454  * main program function to send Comedi data to the MQTT server
00455  */
00456 void comedi_push_mqtt(void)
00457 {
00458     mqtt_bmc_data(E.client_p, TOPIC_P);
00459 }
00460
00461 double ac0_filter(const double raw)
00462 {
00463     static double accum = 0.0f;
00464     ;
00465     static double coef = COEF;
00466     accum = accum - accum / coef + raw;
00467     return accum / coef;
00468 }
00469
00470 double ac1_filter(const double raw)
00471 {
00472     static double accum = 0.0f;
00473     static double coef = COEF;
00474     accum = accum - accum / coef + raw;
00475     return accum / coef;
00476 }

```

4.7 bmc_mqtt.h

```

00001
00002 #ifndef BMC_MQTT_H
00003 #define BMC_MQTT_H
00004
00005 #ifdef __cplusplus
00006 extern "C" {
00007 #endif
00008
00009 #include "bmc.h"
00010 #include "daq.h"
00011 #include "mqtt_rec.h"
00012 #include "mqtt_vars.h"
00013
00014 #define MQTT_RETRY 10
00015
00016     extern struct ha_flag_type ha_flag_vars_ss;
00017
00018     void mqtt_bmc_data(MQTTClient, const char *);
00019     void delivered(void *, MQTTClient_deliveryToken);
00020     int32_t msgarrvd(void *, char *, int, MQTTClient_message *);
00021     void connlost(void *, char *);
00022     void showIP(void);
00023     void skeleton_daemon(void);
00024     void bmc_mqtt_init(void);
00025     char * log_time(bool);
00026
00027     void timer_callback(int32_t);
00028     void comedi_push_mqtt(void);
00029
00030 #ifdef __cplusplus
00031 }

```

```
00032 #endif
00033
00034 #endif /* BMC_MQTT_H */
00035
```

4.8 bmc.o.d

```
00001 build/Debug/GNU-Linux/bmc.o: bmc.c daq.h bmc.h mqtt_rec.h mqtt_vars.h \
00002   bmc_mqtt.h
00003 daq.h:
00004 bmc.h:
00005 mqtt_rec.h:
00006 mqtt_vars.h:
00007 bmc_mqtt.h:
```

4.9 bmc.o.d

```
00001 build/Release/GNU-Linux/bmc.o: bmc.c daq.h bmc.h mqtt_rec.h mqtt_vars.h \
00002   bmc_mqtt.h
00003 daq.h:
00004 bmc.h:
00005 mqtt_rec.h:
00006 mqtt_vars.h:
00007 bmc_mqtt.h:
```

4.10 bmc_mqtt.o.d

```
00001 build/Debug/GNU-Linux/bmc_mqtt.o: bmc_mqtt.c bmc_mqtt.h bmc.h daq.h \
00002   mqtt_rec.h mqtt_vars.h
00003 bmc_mqtt.h:
00004 bmc.h:
00005 daq.h:
00006 mqtt_rec.h:
00007 mqtt_vars.h:
```

4.11 bmc_mqtt.o.d

```
00001 build/Release/GNU-Linux/bmc_mqtt.o: bmc_mqtt.c bmc_mqtt.h bmc.h daq.h \
00002   mqtt_rec.h mqtt_vars.h
00003 bmc_mqtt.h:
00004 bmc.h:
00005 daq.h:
00006 mqtt_rec.h:
00007 mqtt_vars.h:
```

4.12 daq.o.d

```
00001 build/Debug/GNU-Linux/daq.o: daq.c daq.h bmc.h
00002 daq.h:
00003 bmc.h:
```

4.13 daq.o.d

```
00001 build/Release/GNU-Linux/daq.o: daq.c daq.h bmc.h
00002 daq.h:
00003 bmc.h:
```

4.14 daq.c

```

00001 /*
00002  * \file daq.c
00003  */
00004
00005
00006
00007
00008 #include <stdio.h> /* for printf() */
00009 #include <unistd.h>
00010 #include <stdbool.h>
00011 #include <stdint.h>
00012 #include <comedilib.h>
00013 #include "daq.h"
00014
00015 int subdev_ai = 0; /* change this to your input subdevice */
00016 int chan_ai = 0; /* change this to your channel */
00017 int range_ai = 0; /* more on this later */
00018 int aref_ai = AREF_GROUND; /* more on this later */
00019 int maxdata_ai, ranges_ai, channels_ai;
00020
00021 int subdev_ao = 0; /* change this to your input subdevice */
00022 int chan_ao = 0; /* change this to your channel */
00023 int range_ao = 0; /* more on this later */
00024 int aref_ao = AREF_GROUND; /* more on this later */
00025 int maxdata_ao, ranges_ao, channels_ao;
00026
00027 int subdev_di = 0; /* change this to your input subdevice */
00028 int chan_di = 0; /* change this to your channel */
00029 int range_di = 0; /* more on this later */
00030 int maxdata_di, ranges_di, channels_di, datain_di;
00031
00032 int subdev_do = 0; /* change this to your input subdevice */
00033 int chan_do = 0; /* change this to your channel */
00034 int range_do = 0; /* more on this later */
00035 int maxdata_do, ranges_do, channels_do, datain_do;
00036
00037 int subdev_dio; /* change this to your input subdevice */
00038 int chan_dio = 0; /* change this to your channel */
00039 int range_dio = 0; /* more on this later */
00040 int maxdata_dio, ranges_dio, channels_dio, datain_dio;
00041 int aref_dio; /* more on this later */
00042
00043 int subdev_counter; /* change this to your input subdevice */
00044 int chan_counter = 0; /* change this to your channel */
00045 int range_counter = 0; /* more on this later */
00046 int maxdata_counter, ranges_counter, channels_counter, datain_counter;
00047
00048 comedi_t *it;
00049 comedi_range *ad_range, *da_range;
00050 bool ADC_OPEN = true, DIO_OPEN = true, ADC_ERROR = false, DEV_OPEN = true,
00051      DIO_ERROR = false, HAS_AO = false, DAC_ERROR = false, PWM_OPEN = true,
00052      PWM_ERROR = false;
00053
00054 bool DO_OPEN = true, DI_OPEN = true, DO_ERROR = false;
00055
00056 int init_daq(double min_range, double max_range, int range_update)
00057 {
00058     int i = 0;
00059
00060     it = comedi_open("/dev/comedi0");
00061     if (it == NULL) {
00062         comedi_perror("comedi_open");
00063         DEV_OPEN = false;
00064         return -1;
00065     }
00066
00067     subdev_ai = comedi_find_subdevice_by_type(it, COMEDI_SUBD_AI, subdev_ai);
00068     if (subdev_ai < 0) {
00069         ADC_OPEN = false;
00070     }
00071
00072     subdev_ao = comedi_find_subdevice_by_type(it, COMEDI_SUBD_AO, subdev_ao);
00073     if (subdev_ao < 0) {
00074         HAS_AO = false;
00075     } else {
00076         HAS_AO = true;
00077     }
00078
00079     fprintf(fout, "Comedi DAQ Board Name: %s, Driver: %s\r\n", comedi_get_board_name(it),
00080            comedi_get_driver_name(it));
00081
00082     fprintf(fout, "Subdev AI  %i ", subdev_ai);
00083     channels_ai = comedi_get_n_channels(it, subdev_ai);

```

```

00084     fprintf(fout, "Analog Channels %i ", channels_ai);
00085     maxdata_ai = comedi_get_maxdata(it, subdev_ai, i);
00086     fprintf(fout, "Maxdata %i ", maxdata_ai);
00087     ranges_ai = comedi_get_n_ranges(it, subdev_ai, i);
00088     fprintf(fout, "Ranges %i ", ranges_ai);
00089     ad_range = comedi_get_range(it, subdev_ai, i, range_ai);
00090     if (range_update) {
00091         ad_range->min = min_range;
00092         ad_range->max = max_range;
00093     }
00094     fprintf(fout, ": ad_range .min = %.3f, max = %.3f\r\n", ad_range->min,
00095             ad_range->max);
00096
00097     if (HAS_AO) {
00098         fprintf(fout, "Subdev AO %i ", subdev_ao);
00099         channels_ao = comedi_get_n_channels(it, subdev_ao);
00100         fprintf(fout, "Analog Channels %i ", channels_ao);
00101         maxdata_ao = comedi_get_maxdata(it, subdev_ao, i);
00102         fprintf(fout, "Maxdata %i ", maxdata_ao);
00103         ranges_ao = comedi_get_n_ranges(it, subdev_ao, i);
00104         fprintf(fout, "Ranges %i ", ranges_ao);
00105         da_range = comedi_get_range(it, subdev_ao, i, range_ao);
00106         fprintf(fout, ": da_range .min = %.3f, max = %.3f\r\n", da_range->min,
00107                 da_range->max);
00108     }
00109
00110     ADC_OPEN = true;
00111     comedi_set_global_oor_behavior(COMEDI_OOR_NUMBER);
00112     return 0;
00113 }
00114
00115 int init_dac(double min_range, double max_range, int range_update)
00116 {
00117     int i = 0;
00118
00119     if (!DEV_OPEN) {
00120         it = comedi_open("/dev/comedi0");
00121         if (it == NULL) {
00122             comedi_perror("comedi_open");
00123             ADC_OPEN = false;
00124             DEV_OPEN = false;
00125             return -1;
00126         }
00127         DEV_OPEN = true;
00128     }
00129
00130     subdev_ao = comedi_find_subdevice_by_type(it, COMEDI_SUBD_AO, subdev_ao);
00131     if (subdev_ao < 0) {
00132         HAS_AO = false;
00133     } else {
00134         HAS_AO = true;
00135     }
00136
00137     if (HAS_AO) {
00138         fprintf(fout, "Subdev AO %i ", subdev_ao);
00139         channels_ao = comedi_get_n_channels(it, subdev_ao);
00140         fprintf(fout, "Analog Channels %i ", channels_ao);
00141         maxdata_ao = comedi_get_maxdata(it, subdev_ao, i);
00142         fprintf(fout, "Maxdata %i ", maxdata_ao);
00143         ranges_ao = comedi_get_n_ranges(it, subdev_ao, i);
00144         fprintf(fout, "Ranges %i ", ranges_ao);
00145         da_range = comedi_get_range(it, subdev_ao, i, range_ao);
00146         fprintf(fout, ": da_range .min = %.3f, max = %.3f\r\n", da_range->min,
00147                 da_range->max);
00148     }
00149
00150     comedi_set_global_oor_behavior(COMEDI_OOR_NUMBER);
00151     return 0;
00152 }
00153
00154 int adc_range(double min_range, double max_range)
00155 {
00156     if (ADC_OPEN) {
00157         ad_range->min = min_range;
00158         ad_range->max = max_range;
00159         return 0;
00160     } else {
00161         return -1;
00162     }
00163 }
00164
00165 int dac_range(double min_range, double max_range)
00166 {
00167     if (ADC_OPEN) {
00168         da_range->min = min_range;
00169         da_range->max = max_range;
00170         return 0;

```

```
00171     } else {
00172         return -1;
00173     }
00174 }
00175
00176 int set_dac_volts(int chan, double voltage)
00177 {
00178     lsampl_t data;
00179     int retval;
00180
00181     data = comedi_from_phys(voltage, da_range, maxdata_ao);
00182     bmc.dac_sample[chan] = data;
00183     retval = comedi_data_write(it, subdev_ao, chan, range_ao, aref_ao, data);
00184     if (retval < 0) {
00185         comedi_perror("comedi_data_write in set_dac_volts");
00186         DAC_ERROR = true;
00187     }
00188     return retval;
00189 }
00190
00191 int set_dac_raw(int chan, lsampl_t voltage)
00192 {
00193     int retval;
00194
00195     retval = comedi_data_write(it, subdev_ao, chan, range_ao, aref_ao, voltage);
00196     if (retval < 0) {
00197         comedi_perror("comedi_data_write in set_dac_raw");
00198         DAC_ERROR = true;
00199     }
00200     return retval;
00201 }
00202
00203 double get_adc_volts(int chan)
00204 {
00205     lsampl_t data[16];
00206     int retval;
00207
00208     retval = comedi_data_read_n(it, subdev_ai, chan, range_ai, aref_ai, &data[0], 8);
00209     if (retval < 0) {
00210         comedi_perror("comedi_data_read in get_adc_volts");
00211         ADC_ERROR = true;
00212         return 0.0;
00213     }
00214     bmc.adc_sample[chan] = data[0];
00215
00216     ad_range->min = 0.0f;
00217     ad_range->max = HV_SCALE;
00218
00219     return comedi_to_phys(data[0], ad_range, maxdata_ai);
00220 }
00221
00222 int set_dio_output(int chan)
00223 {
00224     return comedi_dio_config(it,
00225         subdev_dio,
00226         chan,
00227         COMEDI_OUTPUT);
00228 }
00229
00230 int set_dio_input(int chan)
00231 {
00232     return comedi_dio_config(it,
00233         subdev_dio,
00234         chan,
00235         COMEDI_INPUT);
00236 }
00237
00238 int get_dio_bit(int chan)
00239 {
00240     lsampl_t data;
00241     int retval;
00242
00243     retval = comedi_data_read(it, subdev_di, chan, range_di, aref_dio, &data);
00244     if (retval < 0) {
00245         comedi_perror("comedi_data_read in get_dio_bits");
00246         DIO_ERROR = true;
00247         return 0;
00248     }
00249     return data;
00250 }
00251
00252 int put_dio_bit(int chan, int bit_data)
00253 {
00254     lsampl_t data = bit_data;
00255     int retval;
00256
00257     retval = comedi_data_write(it, subdev_do, chan, range_do, aref_dio, data);
```

```

00258     if (retval < 0) {
00259         comedi_perror("comedi_data_write in put_dio_bits");
00260         DIO_ERROR = true;
00261         return -1;
00262     }
00263     return 0;
00264 }
00265
00266 int init_dio(void)
00267 {
00268     int i = 0;
00269
00270     if (!DEV_OPEN) {
00271         it = comedi_open("/dev/comedi0");
00272         if (it == NULL) {
00273             comedi_perror("comedi_open");
00274             DIO_OPEN = false;
00275             DEV_OPEN = false;
00276             return -1;
00277         }
00278         DEV_OPEN = true;
00279     }
00280
00281     subdev_di = comedi_find_subdevice_by_type(it, COMEDI_SUBD_DI, subdev_di);
00282     if (subdev_di < 0) {
00283         DI_OPEN = false;
00284     }
00285     subdev_do = comedi_find_subdevice_by_type(it, COMEDI_SUBD_DO, subdev_do);
00286     if (subdev_do < 0) {
00287         DO_OPEN = false;
00288     }
00289
00290     subdev_dio = comedi_find_subdevice_by_type(it, COMEDI_SUBD_DIO, subdev_dio);
00291     if (subdev_dio < 0) {
00292         DIO_OPEN = false;
00293     }
00294
00295     subdev_counter = comedi_find_subdevice_by_type(it, COMEDI_SUBD_COUNTER, subdev_counter);
00296     if (subdev_counter < 0) {
00297         PWM_OPEN = false;
00298     }
00299
00300     if (DI_OPEN) {
00301         fprintf(fout, "Subdev DI %i ", subdev_di);
00302         channels_di = comedi_get_n_channels(it, subdev_di);
00303         fprintf(fout, "Digital Channels %i ", channels_di);
00304         maxdata_di = comedi_get_maxdata(it, subdev_di, i);
00305         fprintf(fout, "Maxdata %i ", maxdata_di);
00306         ranges_di = comedi_get_n_ranges(it, subdev_di, i);
00307         fprintf(fout, "Ranges %i \r\n", ranges_di);
00308     }
00309
00310     if (DO_OPEN) {
00311         fprintf(fout, "Subdev DO %i ", subdev_do);
00312         channels_do = comedi_get_n_channels(it, subdev_do);
00313         fprintf(fout, "Digital Channels %i ", channels_do);
00314         maxdata_do = comedi_get_maxdata(it, subdev_do, i);
00315         fprintf(fout, "Maxdata %i ", maxdata_do);
00316         ranges_do = comedi_get_n_ranges(it, subdev_do, i);
00317         fprintf(fout, "Ranges %i \r\n", ranges_do);
00318     }
00319
00320     if (DIO_OPEN) {
00321         fprintf(fout, "Subdev DIO %i ", subdev_dio);
00322         channels_dio = comedi_get_n_channels(it, subdev_dio);
00323         fprintf(fout, "Digital Channels %i ", channels_dio);
00324         maxdata_dio = comedi_get_maxdata(it, subdev_dio, i);
00325         fprintf(fout, "Maxdata %i ", maxdata_dio);
00326         ranges_dio = comedi_get_n_ranges(it, subdev_dio, i);
00327         fprintf(fout, "Ranges %i \r\n", ranges_dio);
00328     }
00329
00330     if (PWM_OPEN) {
00331         fprintf(fout, "Subdev COU %i ", subdev_counter);
00332         channels_counter = comedi_get_n_channels(it, subdev_counter);
00333         fprintf(fout, "Digital Channels %i ", channels_counter);
00334         maxdata_counter = comedi_get_maxdata(it, subdev_counter, i);
00335         fprintf(fout, "Maxdata %i ", maxdata_counter);
00336         ranges_counter = comedi_get_n_ranges(it, subdev_counter, i);
00337         fprintf(fout, "Ranges %i \r\n", ranges_counter);
00338     }
00339     return 0;
00340 }
00341
00342 int get_data_sample(void)
00343 {
00344     unsigned int obits;

```

```

00345
00346     bmc.datain.D0 = get_dio_bit(0);
00347
00348     if (JUST_BITS) { // send I/O bit by bit
00349         put_dio_bit(0, bmc.dataout.d.D0);
00350         put_dio_bit(1, bmc.dataout.d.D1);
00351         put_dio_bit(2, bmc.dataout.d.D2);
00352         put_dio_bit(3, bmc.dataout.d.D3);
00353         put_dio_bit(4, bmc.dataout.d.D4);
00354         put_dio_bit(5, bmc.dataout.d.D5);
00355         put_dio_bit(6, bmc.dataout.d.D6);
00356         put_dio_bit(7, bmc.dataout.d.D7);
00357     } else { // send I/O as a byte mask
00358         obits = bmc.dataout.dio_buf;
00359         comedi_dio_bitfield2(it, subdev_do, 0xff, &obits, 0);
00360     }
00361
00362     return 0;
00363 }
00364
00365 double lp_filter(double new, int bn, int slow) // low pass filter, slow rate of change for new,
LPCHANC channels, slow/fast select (-1) to zero channel
00366 {
00367     static double smooth[LPCHANC] = {0};
00368     double lp_speed, lp_x;
00369
00370     if ((bn >= LPCHANC) || (bn < 0)) // check for proper array position
00371         return new;
00372     if (slow) {
00373         lp_speed = 0.033;
00374     } else {
00375         lp_speed = 0.125;
00376     }
00377     lp_x = ((smooth[bn]*100.0) + (((new * 100.0)-(smooth[bn]*100.0)) * lp_speed)) / 100.0;
00378     smooth[bn] = lp_x;
00379     if (slow == (-1)) { // reset and return zero
00380         lp_x = 0.0;
00381         smooth[bn] = 0.0;
00382     }
00383     return lp_x;
00384 }

```

4.15 daq.h

```

00001 /*
00002  * File:   daq.h
00003  * Author: root
00004  *
00005  * Created on September 21, 2012, 6:49 PM
00006  */
00007
00008 #ifndef DAQ_H
00009 #define DAQ_H
00010
00011 #ifdef __cplusplus
00012 extern "C" {
00013 #endif
00014
00015 #define PVV_C    0
00016 #define CCV_C    1
00017 #define SYV_C    2
00018 #define B1V_C    3
00019 #define B2V_C    4
00020 #define INV_C    5
00021 #define VD5_C    7
00022 #define PVC_C    8
00023 #define CCC_C    9
00024 #define BAC_C    10
00025
00026 #define LPCHANC    16
00027
00028 #define JUST_BITS false
00029
00030 #define HV_SCALE    83.6f
00031
00032 #include <stdint.h>
00033 #include <comedilib.h>
00034 #include "bmc.h"
00035
00036     struct didata {
00037         uint32_t D0 : 1; //
00038         uint32_t D1 : 1; //
00039         uint32_t D2 : 1; //

```



```

00040         uint32_t D3 : 1; //
00041         uint32_t D4 : 1; //
00042         uint32_t D5 : 1; //
00043         uint32_t D6 : 1; //
00044         uint32_t D7 : 1; //
00045     };
00046
00047     union dio_buf_type {
00048         uint32_t dio_buf;
00049         struct didata d;
00050     };
00051
00052     typedef struct bmcdata {
00053         double pv_voltage, cc_voltage, input_voltage, b1_voltage, b2_voltage, system_voltage,
logic_voltage;
00054         double pv_current, cc_current, battery_current;
00055         struct didata datain;
00056         union dio_buf_type dataout;
00057         int32_t adc_sample[32];
00058         int32_t dac_sample[32];
00059         int32_t utc;
00060     }
00061     bmctype;
00062
00063     extern volatile struct bmcdata bmc;
00064     extern struct didata datain;
00065     extern struct dodata dataout;
00066
00067     extern int maxdata_ai, ranges_ai, channels_ai;
00068     extern int maxdata_ao, ranges_ao, channels_ao;
00069     extern int maxdata_di, ranges_di, channels_di, datain_di;
00070     extern int maxdata_do, ranges_do, channels_do, datain_do;
00071     extern int maxdata_counter, ranges_counter, channels_counter, datain_counter;
00072
00073     int init_daq(double, double, int);
00074     int init_dac(double, double, int);
00075     int init_dio(void);
00076     int adc_range(double, double);
00077     int dac_range(double, double);
00078     double get_adc_volts(int);
00079     int set_dac_volts(int, double);
00080     int set_dac_raw(int, lsampl_t);
00081     int get_dio_bit(int);
00082     int put_dio_bit(int, int);
00083     int set_dio_input(int);
00084     int set_dio_output(int);
00085     int get_data_sample(void);
00086     double lp_filter(double, int, int);
00087 #ifdef __cplusplus
00088 }
00089 #endif
00090
00091 #endif /* DAQ_H */
00092

```

4.16 mqtt_rec.h

```

00001
00002
00003 #ifndef MQTT_REC_H
00004 #define MQTT_REC_H
00005
00006 #ifdef __cplusplus
00007 extern "C" {
00008 #endif
00009
00010 #include "mqtt_vars.h"
00011
00012 #define RDEV_SIZE      10
00013
00014 #define SLEEP_CODE     0
00015 #define FLOAT_CODE     1
00016     //#define DEBUG_REC
00017     //#define GET_DEBUG
00018
00019 #define MBMQTT  1024
00020
00021     enum mqtt_id {
00022         P8055_ID,
00023         FM80_ID,
00024         DUMpload_ID,
00025         HA_ID,
00026         COMEDI_ID,

```

```

00027         LAST_MQTT_ID,
00028     };
00029
00030     struct ha_flag_type {
00031         volatile MQTTClient_deliveryToken deliveredtoken, receivedtoken;
00032         volatile bool runner, rec_ok;
00033         int32_t ha_id;
00034         volatile int32_t var_update, energy_mode;
00035     };
00036
00037     extern FILE* fout;
00038
00039     int32_t msgarrvd(void *, char *, int, MQTTClient_message *);
00040     void delivered(void *, MQTTClient_deliveryToken);
00041
00042     bool json_get_data(cJSON *, const char *, cJSON *, uint32_t);
00043     bool fm80_float(const bool set_bias);
00044     bool fm80_sleep(void);
00045
00046
00047 #ifdef __cplusplus
00048 }
00049 #endif
00050
00051 #endif /* MQTT_REC_H */
00052

```

4.17 mqtt_vars.h

```

00001
00002
00003 #ifndef MQTT_VARS_H
00004 #define MQTT_VARS_H
00005
00006 #ifdef __cplusplus
00007 extern "C" {
00008 #endif
00009
00010 #define HA_SW_DELAY      400000 // usecs
00011 #define TOKEN_DELAY      600
00012 #define GTI_TOKEN_DELAY  300
00013
00014 #define MAIN_DELAY       1000 // 1msec comedi sample rate max
00015
00016 #define QOS               1
00017
00018     void mqtt_ha_switch(MQTTClient, const char *, const bool);
00019     void mqtt_ha_pid(MQTTClient, const char *);
00020     void mqtt_ha_shutdown(MQTTClient, const char *);
00021     bool mqtt_gti_power(MQTTClient, const char *, char *, uint32_t);
00022     bool mqtt_gti_time(MQTTClient, const char *, char *);
00023
00024
00025 #ifdef __cplusplus
00026 }
00027 #endif
00028
00029 #endif /* MQTT_VARS_H */
00030

```

4.18 c_standard_headers_indexer.c

```

00001 /*
00002  * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
00003  *
00004  * Copyright (c) 2016 Oracle and/or its affiliates. All rights reserved.
00005  *
00006  * Oracle and Java are registered trademarks of Oracle and/or its affiliates.
00007  * Other names may be trademarks of their respective owners.
00008  *
00009  * The contents of this file are subject to the terms of either the GNU
00010  * General Public License Version 2 only ("GPL") or the Common
00011  * Development and Distribution License("CDDL") (collectively, the
00012  * "License"). You may not use this file except in compliance with the
00013  * License. You can obtain a copy of the License at
00014  * http://www.netbeans.org/cddl-gplv2.html
00015  * or nbbuild/licenses/CDDL-GPL-2-CP. See the License for the
00016  * specific language governing permissions and limitations under the

```

```

00017  * License. When distributing the software, include this License Header
00018  * Notice in each file and include the License file at
00019  * nbbuild/licenses/CDDL-GPL-2-CP. Oracle designates this
00020  * particular file as subject to the "Classpath" exception as provided
00021  * by Oracle in the GPL Version 2 section of the License file that
00022  * accompanied this code. If applicable, add the following below the
00023  * License Header, with the fields enclosed by brackets [] replaced by
00024  * your own identifying information:
00025  * "Portions Copyrighted [year] [name of copyright owner]"
00026  *
00027  * If you wish your version of this file to be governed by only the CDDL
00028  * or only the GPL Version 2, indicate your decision by adding
00029  * "[Contributor] elects to include this software in this distribution
00030  * under the [CDDL or GPL Version 2] license." If you do not indicate a
00031  * single choice of license, a recipient has the option to distribute
00032  * your version of this file under either the CDDL, the GPL Version 2 or
00033  * to extend the choice of license to its licensees as provided above.
00034  * However, if you add GPL Version 2 code and therefore, elected the GPL
00035  * Version 2 license, then the option applies only if the new code is
00036  * made subject to such option by the copyright holder.
00037  *
00038  * Contributor(s):
00039  */
00040
00041 // List of standard headers was taken in http://en.cppreference.com/w/c/header
00042
00043 #include <assert.h> // Conditionally compiled macro that compares its argument to zero
00044 #include <ctype.h> // Functions to determine the type contained in character data
00045 #include <errno.h> // Macros reporting error conditions
00046 #include <float.h> // Limits of float types
00047 #include <limits.h> // Sizes of basic types
00048 #include <locale.h> // Localization utilities
00049 #include <math.h> // Common mathematics functions
00050 #include <setjmp.h> // Nonlocal jumps
00051 #include <signal.h> // Signal handling
00052 #include <stdarg.h> // Variable arguments
00053 #include <stddef.h> // Common macro definitions
00054 #include <stdio.h> // Input/output
00055 #include <string.h> // String handling
00056 #include <stdlib.h> // General utilities: memory management, program utilities, string
    conversions, random numbers
00057 #include <time.h> // Time/date utilities
00058 #include <iso646.h> // (since C95) Alternative operator spellings
00059 #include <wchar.h> // (since C95) Extended multibyte and wide character utilities
00060 #include <wctype.h> // (since C95) Wide character classification and mapping utilities
00061 #ifdef _STDC_C99
00062 #include <complex.h> // (since C99) Complex number arithmetic
00063 #include <fenv.h> // (since C99) Floating-point environment
00064 #include <inttypes.h> // (since C99) Format conversion of integer types
00065 #include <stdbool.h> // (since C99) Boolean type
00066 #include <stdint.h> // (since C99) Fixed-width integer types
00067 #include <tgmath.h> // (since C99) Type-generic math (macros wrapping math.h and complex.h)
00068 #endif
00069 #ifdef _STDC_C11
00070 #include <stdalign.h> // (since C11) alignas and alignof convenience macros
00071 #include <stdatomic.h> // (since C11) Atomic types
00072 #include <stdnoreturn.h> // (since C11) noreturn convenience macros
00073 #include <threads.h> // (since C11) Thread library
00074 #include <uchar.h> // (since C11) UTF-16 and UTF-32 character utilities
00075 #endif

```

4.19 cpp_standard_headers_indexer.cpp

```

00001  /*
00002  * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
00003  *
00004  * Copyright (c) 2016 Oracle and/or its affiliates. All rights reserved.
00005  *
00006  * Oracle and Java are registered trademarks of Oracle and/or its affiliates.
00007  * Other names may be trademarks of their respective owners.
00008  *
00009  * The contents of this file are subject to the terms of either the GNU
00010  * General Public License Version 2 only ("GPL") or the Common
00011  * Development and Distribution License("CDDL") (collectively, the
00012  * "License"). You may not use this file except in compliance with the
00013  * License. You can obtain a copy of the License at
00014  * http://www.netbeans.org/cddl-gplv2.html
00015  * or nbbuild/licenses/CDDL-GPL-2-CP. See the License for the
00016  * specific language governing permissions and limitations under the
00017  * License. When distributing the software, include this License Header
00018  * Notice in each file and include the License file at
00019  * nbbuild/licenses/CDDL-GPL-2-CP. Oracle designates this
00020  * particular file as subject to the "Classpath" exception as provided

```

```

00021 * by Oracle in the GPL Version 2 section of the License file that
00022 * accompanied this code. If applicable, add the following below the
00023 * License Header, with the fields enclosed by brackets [] replaced by
00024 * your own identifying information:
00025 * "Portions Copyrighted [year] [name of copyright owner]"
00026 *
00027 * If you wish your version of this file to be governed by only the CDDL
00028 * or only the GPL Version 2, indicate your decision by adding
00029 * "[Contributor] elects to include this software in this distribution
00030 * under the [CDDL or GPL Version 2] license." If you do not indicate a
00031 * single choice of license, a recipient has the option to distribute
00032 * your version of this file under either the CDDL, the GPL Version 2 or
00033 * to extend the choice of license to its licensees as provided above.
00034 * However, if you add GPL Version 2 code and therefore, elected the GPL
00035 * Version 2 license, then the option applies only if the new code is
00036 * made subject to such option by the copyright holder.
00037 *
00038 * Contributor(s):
00039 */
00040
00041 // List of standard headers was taken in http://en.cppreference.com/w/cpp/header
00042
00043 #include <cstdlib>          // General purpose utilities: program control, dynamic memory allocation,
                             // random numbers, sort and search
00044 #include <csignal>          // Functions and macro constants for signal management
00045 #include <csetjmp>          // Macro (and function) that saves (and jumps) to an execution context
00046 #include <csdarg>          // Handling of variable length argument lists
00047 #include <typeinfo>        // Runtime type information utilities
00048 #include <bitset>          // std::bitset class template
00049 #include <functional>       // Function objects, designed for use with the standard algorithms
00050 #include <utility>          // Various utility components
00051 #include <ctime>            // C-style time/date utilities
00052 #include <cstdint>          // typedefs for types such as size_t, NULL and others
00053 #include <new>              // Low-level memory management utilities
00054 #include <memory>          // Higher level memory management utilities
00055 #include <climits>         // limits of integral types
00056 #include <cmath>           // limits of float types
00057 #include <limits>          // standardized way to query properties of arithmetic types
00058 #include <exception>       // Exception handling utilities
00059 #include <stdexcept>       // Standard exception objects
00060 #include <cassert>         // Conditionally compiled macro that compares its argument to zero
00061 #include <cerrno>          // Macro containing the last error number
00062 #include <cctype>          // functions to determine the type contained in character data
00063 #include <cwctype>         // functions for determining the type of wide character data
00064 #include <cstring>         // various narrow character string handling functions
00065 #include <cwchar>          // various wide and multibyte string handling functions
00066 #include <string>          // std::basic_string class template
00067 #include <vector>           // std::vector container
00068 #include <deque>           // std::deque container
00069 #include <list>            // std::list container
00070 #include <set>              // std::set and std::multiset associative containers
00071 #include <map>             // std::map and std::multimap associative containers
00072 #include <stack>           // std::stack container adaptor
00073 #include <queue>           // std::queue and std::priority_queue container adaptors
00074 #include <algorithm>       // Algorithms that operate on containers
00075 #include <iterator>        // Container iterators
00076 #include <cmath>           // Common mathematics functions
00077 #include <complex>         // Complex number type
00078 #include <valarray>        // Class for representing and manipulating arrays of values
00079 #include <numeric>         // Numeric operations on values in containers
00080 #include <iosfwd>          // forward declarations of all classes in the input/output library
00081 #include <ios>             // std::ios_base class, std::basic_ios class template and several typedefs
00082 #include <istream>         // std::basic_istream class template and several typedefs
00083 #include <ostream>        // std::basic_ostream, std::basic_iostream class templates and several
                             // typedefs
00084 #include <iostream>        // several standard stream objects
00085 #include <fstream>         // std::basic_fstream, std::basic_ifstream, std::basic_ofstream class
                             // templates and several typedefs
00086 #include <sstream>         // std::basic_stringstream, std::basic_istringstream,
                             // std::basic_ostringstream class templates and several typedefs
00087 #include <strstream>       // std::strstream, std::istrstream, std::ostrstream(deprecated)
00088 #include <iomanip>         // Helper functions to control the format or input and output
00089 #include <streambuf>       // std::basic_streambuf class template
00090 #include <cstdio>          // C-style input-output functions
00091 #include <locale>          // Localization utilities
00092 #include <locale>          // C localization utilities
00093 #include <ciso646>         // empty header. The macros that appear in iso646.h in C are keywords in
                             // C++
00094 #if __cplusplus >= 201103L
00095 #include <type_index>      // (since C++11)    std::type_index
00096 #include <type_traits>     // (since C++11)    Compile-time type information
00097 #include <chrono>          // (since C++11)    C++ time utilities
00098 #include <initializer_list> // (since C++11)    std::initializer_list class template
00099 #include <tuple>           // (since C++11)    std::tuple class template
00100 #include <scoped_allocator> // (since C++11)    Nested allocator class
00101 #include <cstdint>         // (since C++11)    fixed-size types and limits of other types
00102 #include <cinttypes>       // (since C++11)    formatting macros , intmax_t and uintmax_t math and

```

```

        conversions
00103 #include <system_error>    // (since C++11)    defines std::error_code, a platform-dependent error
        code
00104 #include <cuchar>           // (since C++11)    C-style Unicode character conversion functions
00105 #include <array>            // (since C++11)    std::array container
00106 #include <forward_list>     // (since C++11)    std::forward_list container
00107 #include <unordered_set>     // (since C++11)    std::unordered_set and std::unordered_multiset
        unordered associative containers
00108 #include <unordered_map>     // (since C++11)    std::unordered_map and std::unordered_multimap
        unordered associative containers
00109 #include <random>           // (since C++11)    Random number generators and distributions
00110 #include <ratio>            // (since C++11)    Compile-time rational arithmetic
00111 #include <cfenv>            // (since C++11)    Floating-point environment access functions
00112 #include <codecvt>          // (since C++11)    Unicode conversion facilities
00113 #include <regex>            // (since C++11)    Classes, algorithms and iterators to support regular
        expression processing
00114 #include <atomic>           // (since C++11)    Atomic operations library
00115 #include <ccomplex>         // (since C++11) (deprecated in C++17)    simply includes the header
        <complex>
00116 #include <ctgmath>          // (since C++11) (deprecated in C++17)    simply includes the headers
        <ccomplex> (until C++17) <complex> (since C++17) and <cmath>: the overloads equivalent to the contents
        of the C header tgmath.h are already provided by those headers
00117 #include <cstdalign>        // (since C++11) (deprecated in C++17)    defines one compatibility macro
        constant
00118 #include <cstdbool>         // (since C++11) (deprecated in C++17)    defines one compatibility macro
        constant
00119 #include <thread>           // (since C++11)    std::thread class and supporting functions
00120 #include <mutex>            // (since C++11)    mutual exclusion primitives
00121 #include <future>           // (since C++11)    primitives for asynchronous computations
00122 #include <condition_variable> // (since C++11)    thread waiting conditions
00123 #endif
00124 #if __cplusplus >= 201300L
00125 #include <shared_mutex>     // (since C++14)    shared mutual exclusion primitives
00126 #endif
00127 #if __cplusplus >= 201500L
00128 #include <any>              // (since C++17)    std::any class template
00129 #include <optional>         // (since C++17)    std::optional class template
00130 #include <variant>          // (since C++17)    std::variant class template
00131 #include <memory_resource>   // (since C++17)    Polymorphic allocators and memory resources
00132 #include <string_view>       // (since C++17)    std::basic_string_view class template
00133 #include <execution>         // (since C++17)    Predefined execution policies for parallel versions of
        the algorithms
00134 #include <filesystem>        // (since C++17)    std::path class and supporting functions
00135 #endif

```


Index

- ac0_filter
 - bmc_mqtt.c, 26
- ac1_filter
 - bmc_mqtt.c, 26
- ac_low_adj
 - energy_type, 12
- ac_mismatch
 - energy_type, 12
- ac_sw_on
 - energy_type, 12
- ac_sw_status
 - energy_type, 12
- adc
 - energy_type, 12
- adc_sample
 - bmcddata, 6
- b1_voltage
 - bmcddata, 6
- b2_voltage
 - bmcddata, 6
- battery_current
 - bmcddata, 6
- bmc
 - bmc.c, 20
- bmc.c, 19
 - bmc, 20
 - board_name, 20
 - driver_name, 20
 - E, 21
 - fout, 21
 - led_lightshow, 20
 - main, 20
 - sine_wave, 21
- bmc_mqtt.c, 25
 - ac0_filter, 26
 - ac1_filter, 26
 - bmc_mqtt_init, 27
 - COEF, 26
 - comedi_push_mqtt, 27
 - conn_opts_ha, 28
 - conn_opts_p, 28
 - conn_opts_sd, 29
 - connlost, 27
 - delivered, 27
 - ha_flag_vars_ss, 29
 - hname, 29
 - hname_len, 29
 - hname_ptr, 29
 - log_time, 27
 - mqtt_bmc_data, 27
 - msgarrvd, 28
 - new_timer, 29
 - old_timer, 30
 - pubmsg, 30
 - rawtime, 30
 - showIP, 28
 - skeleton_daemon, 28
 - timer_callback, 28
 - token, 30
- bmc_mqtt_init
 - bmc_mqtt.c, 27
- bmcddata, 5
 - adc_sample, 6
 - b1_voltage, 6
 - b2_voltage, 6
 - battery_current, 6
 - cc_current, 6
 - cc_voltage, 6
 - dac_sample, 7
 - datain, 7
 - dataout, 7
 - input_voltage, 7
 - logic_voltage, 7
 - pv_current, 7
 - pv_voltage, 7
 - system_voltage, 7
 - utc, 8
- board_name
 - bmc.c, 20
- build/Debug/GNU-Linux/bmc.o.d, 37
- build/Debug/GNU-Linux/bmc_mqtt.o.d, 37
- build/Debug/GNU-Linux/daq.o.d, 37
- build/Release/GNU-Linux/bmc.o.d, 37
- build/Release/GNU-Linux/bmc_mqtt.o.d, 37
- build/Release/GNU-Linux/daq.o.d, 37
- cc_current
 - bmcddata, 6
- cc_voltage
 - bmcddata, 6
- client_ha
 - energy_type, 12
- client_p
 - energy_type, 12
- client_sd
 - energy_type, 12
- COEF
 - bmc_mqtt.c, 26
- comedi

- energy_type, 13
- comedi_push_mqtt
 - bmc_mqtt.c, 27
- conn_opts_ha
 - bmc_mqtt.c, 28
- conn_opts_p
 - bmc_mqtt.c, 28
- conn_opts_sd
 - bmc_mqtt.c, 29
- connlost
 - bmc_mqtt.c, 27
- d
 - dio_buf_type, 10
- D0
 - didata, 8
- D1
 - didata, 8
- D2
 - didata, 8
- D3
 - didata, 9
- D4
 - didata, 9
- D5
 - didata, 9
- D6
 - didata, 9
- D7
 - didata, 9
- dac
 - energy_type, 13
- dac_sample
 - bmcddata, 7
- datain
 - bmcddata, 7
- dataout
 - bmcddata, 7
- dc_mismatch
 - energy_type, 13
- delivered
 - bmc_mqtt.c, 27
- deliveredtoken
 - ha_flag_type, 17
- di_16b
 - energy_type, 13
- didata, 8
 - D0, 8
 - D1, 8
 - D2, 8
 - D3, 9
 - D4, 9
 - D5, 9
 - D6, 9
 - D7, 9
- dio_buf
 - dio_buf_type, 10
- dio_buf_type, 10
 - d, 10

- dio_buf, 10
- dl_excess
 - energy_type, 13
- dl_excess_adj
 - energy_type, 13
- do_16b
 - energy_type, 13
- driver_name
 - bmc.c, 20
- dumpload
 - energy_type, 13
- E
 - bmc.c, 21
- energy_mode
 - ha_flag_type, 17
- energy_type, 11
 - ac_low_adj, 12
 - ac_mismatch, 12
 - ac_sw_on, 12
 - ac_sw_status, 12
 - adc, 12
 - client_ha, 12
 - client_p, 12
 - client_sd, 12
 - comedi, 13
 - dac, 13
 - dc_mismatch, 13
 - di_16b, 13
 - dl_excess, 13
 - dl_excess_adj, 13
 - do_16b, 13
 - dumpload, 13
 - fm80, 14
 - gti_delay, 14
 - gti_low_adj, 14
 - gti_sw_on, 14
 - gti_sw_status, 14
 - ha_lock, 14
 - homeassistant, 14
 - iammeter, 14
 - im_delay, 15
 - im_display, 15
 - log_spam, 15
 - log_time_reset, 15
 - mode_mismatch, 15
 - mqtt_count, 15
 - once_ac, 15
 - once_gti, 15
 - once_gti_zero, 16
 - rc, 16
 - sane, 16
 - sequence, 16
 - solar_mode, 16
 - solar_shutdown, 16
 - speed_go, 16
 - startup, 16
 - thirty_sec_clock, 17

- fm80
 - energy_type, 14
- fout
 - bmc.c, 21
- gti_delay
 - energy_type, 14
- gti_low_adj
 - energy_type, 14
- gti_sw_on
 - energy_type, 14
- gti_sw_status
 - energy_type, 14
- ha_flag_type, 17
 - deliveredtoken, 17
 - energy_mode, 17
 - ha_id, 17
 - rec_ok, 18
 - receivedtoken, 18
 - runner, 18
 - var_update, 18
- ha_flag_vars_ss
 - bmc_mqtt.c, 29
- ha_id
 - ha_flag_type, 17
- ha_lock
 - energy_type, 14
- hname
 - bmc_mqtt.c, 29
- hname_len
 - bmc_mqtt.c, 29
- hname_ptr
 - bmc_mqtt.c, 29
- homeassistant
 - energy_type, 14
- iammeter
 - energy_type, 14
- im_delay
 - energy_type, 15
- im_display
 - energy_type, 15
- input_voltage
 - bmcddata, 7
- led_lightshow
 - bmc.c, 20
- log_spam
 - energy_type, 15
- log_time
 - bmc_mqtt.c, 27
- log_time_reset
 - energy_type, 15
- logic_voltage
 - bmcddata, 7
- main
 - bmc.c, 20
- mode_mismatch
 - energy_type, 15
- mqtt_bmc_data
 - bmc_mqtt.c, 27
- mqtt_count
 - energy_type, 15
- msgarrvd
 - bmc_mqtt.c, 28
- nbproject/private/c_standard_headers_indexer.c, 44
- nbproject/private/cpp_standard_headers_indexer.cpp, 45
- new_timer
 - bmc_mqtt.c, 29
- old_timer
 - bmc_mqtt.c, 30
- once_ac
 - energy_type, 15
- once_gti
 - energy_type, 15
- once_gti_zero
 - energy_type, 16
- pubmsg
 - bmc_mqtt.c, 30
- pv_current
 - bmcddata, 7
- pv_voltage
 - bmcddata, 7
- rawtime
 - bmc_mqtt.c, 30
- rc
 - energy_type, 16
- rec_ok
 - ha_flag_type, 18
- receivedtoken
 - ha_flag_type, 18
- runner
 - ha_flag_type, 18
- sane
 - energy_type, 16
- sequence
 - energy_type, 16
- showIP
 - bmc_mqtt.c, 28
- sine_wave
 - bmc.c, 21
- skeleton_daemon
 - bmc_mqtt.c, 28
- solar_mode
 - energy_type, 16
- solar_shutdown
 - energy_type, 16
- speed_go
 - energy_type, 16
- startup

- energy_type, [16](#)
- system_voltage
 - bmcdata, [7](#)
- thirty_sec_clock
 - energy_type, [17](#)
- timer_callback
 - bmc_mqtt.c, [28](#)
- token
 - bmc_mqtt.c, [30](#)
- utc
 - bmcdata, [8](#)
- var_update
 - ha_flag_type, [18](#)