My Project

Generated by Doxygen 1.14.0

Chapter 1

Data Structure Index

1.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 2

File Index

2.1 File List

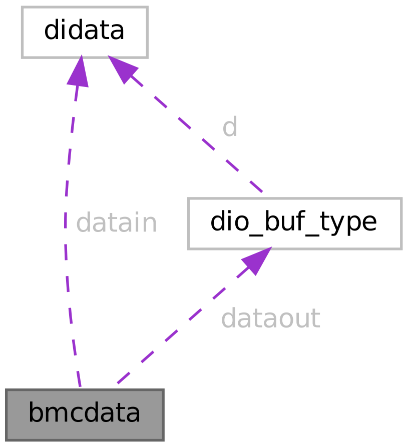
Here is a list of all documented files with brief descriptions:

Chapter 3

Data Structure Documentation

3.1 bmcdata Struct Reference

Collaboration diagram for bmcdata:



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 50 of file daq.h.

3.1.2 Field Documentation

3.1.2.1 adc_sample

```
int32_t adc_sample[32]
```

Definition at line 55 of file daq.h.

3.1.2.2 b1_voltage

```
double b1_voltage
```

Definition at line 51 of file daq.h.

3.1.2.3 b2_voltage

```
double b2_voltage
```

Definition at line 51 of file daq.h.

3.1.2.4 battery_current

```
double battery_current
```

Definition at line 52 of file daq.h.

3.1.2.5 cc_current

```
double cc_current
```

Definition at line 52 of file daq.h.

3.1.2.6 cc_voltage

```
double cc_voltage
```

Definition at line 51 of file daq.h.

3.1.2.7 dac_sample

```
int32_t dac_sample[32]
```

Definition at line 56 of file daq.h.

3.1.2.8 datain

```
struct didata datain
```

Definition at line 53 of file daq.h.

3.1.2.9 dataout

```
union dio_buf_type dataout
```

Definition at line 54 of file daq.h.

3.1.2.10 input_voltage

```
double input_voltage
```

Definition at line 51 of file daq.h.

3.1.2.11 logic_voltage

```
double logic_voltage
```

Definition at line 51 of file daq.h.

3.1.2.12 pv_current

```
double pv_current
```

Definition at line 52 of file daq.h.

3.1.2.13 pv_voltage

```
double pv_voltage
```

Definition at line 51 of file daq.h.

3.1.2.14 system_voltage

```
double system_voltage
```

Definition at line 51 of file daq.h.

3.1.2.15 utc

```
int32_t utc
```

Definition at line 57 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 34 of file daq.h.

3.2.2 Field Documentation

3.2.2.1 D0

```
uint32_t D0
```

Definition at line 35 of file daq.h.

3.2.2.2 D1

```
uint32_t D1
```

Definition at line 36 of file daq.h.

3.2.2.3 D2

```
uint32_t D2
```

Definition at line 37 of file daq.h.

3.2.2.4 D3

```
uint32_t D3
```

Definition at line 38 of file daq.h.

3.2.2.5 D4

```
uint32_t D4
```

Definition at line 39 of file daq.h.

3.2.2.6 D5

```
uint32_t D5
```

Definition at line 40 of file daq.h.

3.2.2.7 D6

```
uint32_t D6
```

Definition at line 41 of file daq.h.

3.2.2.8 D7

```
uint32_t D7
```

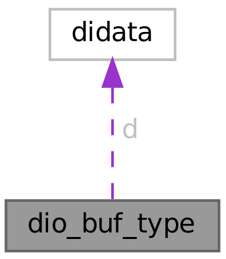
Definition at line 42 of file daq.h.

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 45 of file daq.h.

3.3.2 Field Documentation

3.3.2.1 d

```
struct didata d
```

Definition at line 47 of file daq.h.

3.3.2.2 dio_buf

```
uint32_t dio_buf
```

Definition at line 46 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 ten_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 82 of file bmc.h.

3.4.2 Field Documentation

3.4.2.1 ac_low_adj

```
volatile double ac_low_adj
```

Definition at line 84 of file bmc.h.

3.4.2.2 ac_mismatch

```
volatile bool ac_mismatch
```

Definition at line 85 of file bmc.h.

3.4.2.3 ac_sw_on

```
volatile bool ac_sw_on
```

Definition at line 85 of file bmc.h.

3.4.2.4 ac_sw_status

```
volatile bool ac_sw_status
```

Definition at line 85 of file bmc.h.

3.4.2.5 adc

```
double adc[16]
```

Definition at line 91 of file bmc.h.

3.4.2.6 client_ha

```
MQTTClient client_ha
```

Definition at line 92 of file bmc.h.

3.4.2.7 client_p

```
MQTTClient client_p
```

Definition at line 92 of file bmc.h.

3.4.2.8 client_sd

```
MQTTClient client_sd
```

Definition at line 92 of file bmc.h.

3.4.2.9 comedi

```
volatile bool comedi
```

Definition at line 83 of file bmc.h.

3.4.2.10 dac

```
double dac[16]
```

Definition at line 91 of file bmc.h.

3.4.2.11 dc_mismatch

```
volatile bool dc_mismatch
```

Definition at line 85 of file bmc.h.

3.4.2.12 di_16b

```
volatile int16_t di_16b
```

Definition at line 90 of file bmc.h.

3.4.2.13 dl_excess

```
volatile bool dl_excess
```

Definition at line 85 of file bmc.h.

3.4.2.14 dl_excess_adj

```
volatile double dl_excess_adj
```

Definition at line 84 of file bmc.h.

3.4.2.15 do_16b

```
volatile int16_t do_16b
```

Definition at line 90 of file bmc.h.

3.4.2.16 dumpload

```
volatile bool dumpload
```

Definition at line 83 of file bmc.h.

3.4.2.17 fm80

```
volatile bool fm80
```

Definition at line 83 of file bmc.h.

3.4.2.18 gti_delay

```
volatile uint32_t gti_delay
```

Definition at line 86 of file bmc.h.

3.4.2.19 gti_low_adj

```
volatile double gti_low_adj
```

Definition at line 84 of file bmc.h.

3.4.2.20 gti_sw_on

```
volatile bool gti_sw_on
```

Definition at line 85 of file bmc.h.

3.4.2.21 gti_sw_status

```
volatile bool gti_sw_status
```

Definition at line 85 of file bmc.h.

3.4.2.22 ha_lock

```
pthread_mutex_t ha_lock
```

Definition at line 89 of file bmc.h.

3.4.2.23 homeassistant

```
volatile bool homeassistant
```

Definition at line 83 of file bmc.h.

3.4.2.24 iammeter

```
volatile bool iammeter
```

Definition at line 83 of file bmc.h.

3.4.2.25 im_delay

```
volatile uint32_t im_delay
```

Definition at line 86 of file bmc.h.

3.4.2.26 im_display

```
volatile uint32_t im_display
```

Definition at line 86 of file bmc.h.

3.4.2.27 log_spam

```
volatile uint32_t log_spam
```

Definition at line 88 of file bmc.h.

3.4.2.28 log_time_reset

```
volatile uint32_t log_time_reset
```

Definition at line 88 of file bmc.h.

3.4.2.29 mode_mismatch

```
volatile bool mode_mismatch
```

Definition at line 85 of file bmc.h.

3.4.2.30 mqtt_count

```
volatile uint32_t mqtt_count
```

Definition at line 86 of file bmc.h.

3.4.2.31 once_ac

```
volatile bool once_ac
```

Definition at line 83 of file bmc.h.

3.4.2.32 once_gti

```
volatile bool once_gti
```

Definition at line 83 of file bmc.h.

3.4.2.33 once_gti_zero

```
volatile bool once_gti_zero
```

Definition at line 83 of file bmc.h.

3.4.2.34 rc

```
volatile int32_t rc
```

Definition at line 87 of file bmc.h.

3.4.2.35 sane

```
volatile int32_t sane
```

Definition at line 87 of file bmc.h.

3.4.2.36 sequence

```
volatile uint32_t sequence
```

Definition at line 86 of file bmc.h.

3.4.2.37 solar_mode

```
volatile bool solar_mode
```

Definition at line 85 of file bmc.h.

3.4.2.38 solar_shutdown

```
volatile bool solar_shutdown
```

Definition at line 85 of file bmc.h.

3.4.2.39 speed_go

```
volatile uint32_t speed_go
```

Definition at line 86 of file bmc.h.

3.4.2.40 startup

```
volatile bool startup
```

Definition at line 85 of file bmc.h.

3.4.2.41 ten_sec_clock

```
volatile uint32_t ten_sec_clock
```

Definition at line 88 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

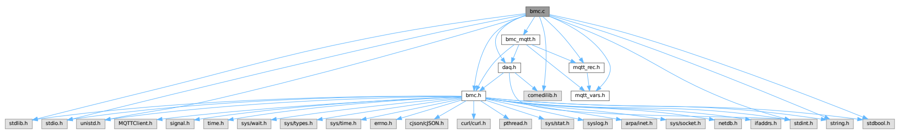
4.1 .dep.inc

```
# This code depends on make tool being used
DEPFILES=$(wildcard $(addsuffix .d, ${OBJECTFILES} ${TESTOBJECTFILES}))
ifneq (${DEPFILES},)
include ${DEPFILES}
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 bmcdata 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 bmcdata 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.

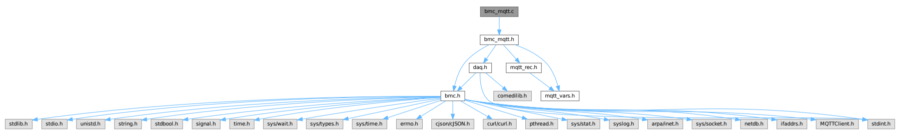
```c

#include <stdlib.h>
#include <stdio.h> /* for printf() */
#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"

volatile struct bmcdata bmc; /* DIO buffer */

struct energy_type E = {
    .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,
};


// Comedi I/O device type
const char *board_name = "NO_BOARD", *driver_name = "NO_DRIVER";

FILE* fout; // logging stream

/* ripped from http://aquaticus.info/pwm-sine-wave */

uint8_t sine_wave[256] = {
    0x80, 0x83, 0x86, 0x89, 0x8C, 0x90, 0x93, 0x96,
    0x99, 0x9C, 0x9F, 0xA2, 0xA5, 0xA8, 0xAB, 0xAE,
    0xB1, 0xB3, 0xB6, 0xB9, 0xBC, 0xBF, 0xC1, 0xC4,
    0xC7, 0xC9, 0xCC, 0xCE, 0xD1, 0xD3, 0xD5, 0xD8,
    0xDA, 0xDC, 0xDE, 0xE0, 0xE2, 0xE4, 0xE6, 0xE8,
    0xEA, 0xEB, 0xED, 0xEF, 0xF0, 0xF1, 0xF3, 0xF4,
    0xF5, 0xF6, 0xF8, 0xF9, 0xFA, 0xFA, 0xFB, 0xFC,
    0xFD, 0xFD, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF,
    0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFE, 0xFD,
    0xFD, 0xFC, 0xFB, 0xFA, 0xFA, 0xF9, 0xF8, 0xF6,
    0xF5, 0xF4, 0xF3, 0xF1, 0xF0, 0xEF, 0xED, 0xEB,
    0xEA, 0xE8, 0xE6, 0xE4, 0xE2, 0xE0, 0xDE, 0xDC,
    0xDA, 0xD8, 0xD5, 0xD3, 0xD1, 0xCE, 0xCC, 0xC9,
    0xC7, 0xC4, 0xC1, 0xBF, 0xBC, 0xB9, 0xB6, 0xB3,
    0xB1, 0xAE, 0xAB, 0xA8, 0xA5, 0xA2, 0x9F, 0x9C,
    0x99, 0x96, 0x93, 0x90, 0x8C, 0x89, 0x86, 0x83,
    0x80, 0x7D, 0x7A, 0x77, 0x74, 0x70, 0x6D, 0x6A,
    0x67, 0x64, 0x61, 0x5E, 0x5B, 0x58, 0x55, 0x52,
    0x4F, 0x4D, 0x4A, 0x47, 0x44, 0x41, 0x3F, 0x3C,
    0x39, 0x37, 0x34, 0x32, 0x2F, 0x2D, 0x2B, 0x28,
    0x26, 0x24, 0x22, 0x20, 0x1E, 0x1C, 0x1A, 0x18,
    0x16, 0x15, 0x13, 0x11, 0x10, 0x0F, 0x0D, 0x0C,
    0x0B, 0x0A, 0x08, 0x07, 0x06, 0x06, 0x05, 0x04,
    0x03, 0x03, 0x02, 0x02, 0x02, 0x01, 0x01, 0x01,
    0x01, 0x01, 0x01, 0x01, 0x02, 0x02, 0x02, 0x03,
    0x03, 0x04, 0x05, 0x06, 0x06, 0x07, 0x08, 0x0A,
    0x0B, 0x0C, 0x0D, 0x0F, 0x10, 0x11, 0x13, 0x15,
    0x16, 0x18, 0x1A, 0x1C, 0x1E, 0x20, 0x22, 0x24,
    0x26, 0x28, 0x2B, 0x2D, 0x2F, 0x32, 0x34, 0x37,
    0x39, 0x3C, 0x3F, 0x41, 0x44, 0x47, 0x4A, 0x4D,
    0x4F, 0x52, 0x55, 0x58, 0x5B, 0x5E, 0x61, 0x64,
    0x67, 0x6A, 0x6D, 0x70, 0x74, 0x77, 0x7A, 0x7D
};

void led_lightshow(int speed)
{
    static int j = 0;
    static uint8_t cylon = 0xff;
    static int alive_led = 0;
    static bool LED_UP = true;

    if (j++ >= speed) { // delay a bit ok
        if (0) { // screen status feedback
            bmc.dataout.dio_buf = ~cylon; // roll leds cylon style
        } else {
            bmc.dataout.dio_buf = cylon; // roll leds cylon style (inverted)
        }

        if (LED_UP && (alive_led != 0)) {
            alive_led = alive_led * 2;
            cylon = cylon « 1;
        } else {
            if (alive_led != 0) alive_led = alive_led / 2;
            cylon = cylon » 1;
        }
        if (alive_led < 2) {
            alive_led = 2;
            LED_UP = true;
        } else {
            if (alive_led > 128) {
                alive_led = 128;
                LED_UP = false;
            }
        }
        j = 0;
    }
}

int main(int argc, char *argv[])
{
    int do_ao_only = false;
    uint8_t i = 0, j = 75;

    /*
     * start the MQTT processing
     */
    bmc_mqtt_init();

    if (do_ao_only) {
        if (init_dac(0.0, 25.0, false) < 0) {
            fprintf(fout, "Missing Analog AO subdevice\n");
            return -1;
        }


        while (true) {
            set_dac_raw(0, sine_wave[255 - i++] « 4);
            set_dac_raw(1, sine_wave[255 - j++] « 4);
        }
    } else {

        if (init_daq(0.0, 25.0, false) < 0) {
            fprintf(fout, "Missing Analog subdevice(s)\n");
            return -1;
        }
        if (init_dio() < 0) {
            fprintf(fout, "Missing Digital subdevice(s)\n");
            return -1;
        }

        //      set_dac_raw(0, 255); // show max Voltage

        E.dac[0] = 2.50f;
        E.dac[1] = 3.33f;

        E.do_16b = 0x01;
        E.di_16b = 0x10;

        fflush(fout);
        while (true) {
            get_data_sample();
            if (!bmc.datain.D0) {
                led_lightshow(10);
            }
            if (ha_flag_vars_ss.runner) { // 30 second timer or trigger from mqtt
                comedi_push_mqtt(); // send json formatted data to the mqtt server
                ha_flag_vars_ss.runner = false;
            }
        }

    }
    return 0;
}


```

4.4 bmc.h

```
/*
 * File:   bmc.h
 * Author: root
 *
 * Created on September 21, 2012, 12:54 PM
 */

#ifndef BMC_H
#define BMC_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdlib.h>
#include <stdio.h> /* for printf() */
#include <unistd.h>
#include <stdint.h>
#include <string.h>
#include <stdbool.h>
#include <signal.h>
#include <time.h>
#include <sys/wait.h>
#include <sys/types.h>
#include <sys/time.h>
#include <errno.h>
#include <cjson/cJSON.h>
#include <curl/curl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <syslog.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <netdb.h>
#include <ifaddrs.h>
#include "MQTTClient.h"

#define LOG_VERSION     "V0.02"
#define MQTT_VERSION    "V3.11"
#define TNAME  "maint9"
#define LADDRESS        "tcp://127.0.0.1:1883"
#ifdef __amd64
#define ADDRESS         "tcp://10.1.1.172:1883"
#else
#define ADDRESS         "tcp://10.1.1.172:1883"
#endif
#define CLIENTID1       "Energy_Mqtt_BMC1"
#define CLIENTID2       "Energy_Mqtt_BMC2"
#define CLIENTID3       "Energy_Mqtt_BMC3"
#define TOPIC_P         "comedi/bmc/data/bmc"
#define TOPIC_SPAM      "comedi/bmc/data/spam"
#define TOPIC_PACA      "home-assistant/comedi/bmc"
#define TOPIC_PACB      "mateq84/data/#"
#define TOPIC_AI        "comedi/bmc/data/ai"
#define TOPIC_AO        "comedi/bmc/data/ao"
#define TOPIC_DI        "comedi/bmc/data/di"
#define TOPIC_DO        "comedi/bmc/data/do"
#define QOS             1

#define TOPIC_SS        "mateq84/data/solar" // receive data testing

#define TIMEOUT         10000L
#define SPACING_USEC    500 * 1000
#define USEC_SEC        1000000L

#define CMD_SEC         30
#define TIME_SYNC_SEC   30

#define SBUF_SIZ        16  // short buffer string size
#define RBUF_SIZ        82
#define SYSLOG_SIZ      512

#define LOG_TO_FILE         "/public/bmc/bmc_comedi.log"
#define LOG_TO_FILE_ALT     "/tmp/bmc_comedi.log"

#define MQTT_RECONN     3
#define KAI             60

        extern FILE* fout; // logging stream
        extern struct energy_type E;

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

        void led_lightshow(int);

#ifdef __cplusplus
}
#endif

#endif /* BMC_H */

```

4.5 bmc_mqtt.c File Reference

`#include "bmc_mqtt.h"`
Include dependency graph for bmc_mqtt.c:



Functions

- void showIP (void)
- void skeleton_daemon (void)
- 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 Function Documentation

4.5.2.1 bmc_mqtt_init()

```
void bmc_mqtt_init (
             void )
```

Definition at line 216 of file bmc_mqtt.c.

4.5.2.2 comedi_push_mqtt()

```
void comedi_push_mqtt (
             void )
```

Definition at line 423 of file bmc_mqtt.c.

4.5.2.3 connlost()

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

trouble in River-city

Definition at line 164 of file bmc_mqtt.c.

4.5.2.4 delivered()

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

Definition at line 202 of file bmc_mqtt.c.

4.5.2.5 log_time()

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

Definition at line 128 of file bmc_mqtt.c.

4.5.2.6 mqtt_bmc_data()

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

Definition at line 354 of file bmc_mqtt.c.

4.5.2.7 msgarrvd()

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

Definition at line 277 of file bmc_mqtt.c.

4.5.2.8 showIP()

```
void showIP (
             void )
```

Definition at line 35 of file bmc_mqtt.c.

4.5.2.9 skeleton_daemon()

```
void skeleton_daemon (
             void )
```

Definition at line 69 of file bmc_mqtt.c.

4.5.2.10 timer_callback()

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

Definition at line 152 of file bmc_mqtt.c.

4.5.3 Variable Documentation

4.5.3.1 conn_opts_ha

```
MQTTClient_connectOptions conn_opts_ha = MQTTClient_connectOptions_initializer
```

Definition at line 16 of file bmc_mqtt.c.

4.5.3.2 conn_opts_p

```
MQTTClient_connectOptions conn_opts_p = MQTTClient_connectOptions_initializer
```

Definition at line 14 of file bmc_mqtt.c.

4.5.3.3 conn_opts_sd

```
MQTTClient_connectOptions conn_opts_sd = MQTTClient_connectOptions_initializer
```

Definition at line 15 of file bmc_mqtt.c.

4.5.3.4 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 22 of file bmc_mqtt.c.

4.5.3.5 hname

```
char hname[256]
```

Definition at line 19 of file bmc_mqtt.c.

4.5.3.6 hname_len

```
size_t hname_len = 12
```

Definition at line 20 of file bmc_mqtt.c.

4.5.3.7 hname_ptr

```
char * hname_ptr = hname
```

Definition at line 19 of file bmc_mqtt.c.

4.5.3.8 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 6 of file bmc_mqtt.c.

4.5.3.9 old_timer

```
struct itimerval old_timer
```

Definition at line 12 of file bmc_mqtt.c.

4.5.3.10 pubmsg

```
MQTTClient_message pubmsg = MQTTClient_message_initializer
```

Definition at line 17 of file bmc_mqtt.c.

4.5.3.11 rawtime

```
time_t rawtime
```

Definition at line 13 of file bmc_mqtt.c.

4.5.3.12 token

```
MQTTClient_deliveryToken token
```

Definition at line 18 of file bmc_mqtt.c.

4.6 bmc_mqtt.c

Go to the documentation of this file.

```c
#include "bmc_mqtt.h"

static const char *const FW_Date = __DATE__;
static const char *const FW_Time = __TIME__;

struct itimerval new_timer = {
    .it_value.tv_sec = CMD_SEC,
    .it_value.tv_usec = 0,
    .it_interval.tv_sec = CMD_SEC,
    .it_interval.tv_usec = 0,
};
struct itimerval old_timer;
time_t rawtime;
MQTTClient_connectOptions conn_opts_p = MQTTClient_connectOptions_initializer,
    conn_opts_sd = MQTTClient_connectOptions_initializer,
    conn_opts_ha = MQTTClient_connectOptions_initializer;
MQTTClient_message pubmsg = MQTTClient_message_initializer;
MQTTClient_deliveryToken token;
char hname[256], *hname_ptr = hname;
size_t hname_len = 12;

struct ha_flag_type ha_flag_vars_ss = {
    .runner = false,
    .receivedtoken = false,
    .deliveredtoken = false,
    .rec_ok = false,
    .ha_id = COMEDI_ID,
    .var_update = 0,
};

void showIP(void)
{
    struct ifaddrs *ifaddr, *ifa;
    int s;
    char host[NI_MAXHOST];

    if (getifaddrs(&ifaddr) == -1) {
        perror("getifaddrs");
        exit(EXIT_FAILURE);
    }


    for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
        if (ifa->ifa_addr == NULL)
            continue;

        s = getnameinfo(ifa->ifa_addr, sizeof(struct sockaddr_in), host, NI_MAXHOST, NULL, 0,
NI_NUMERICHOST);

        if (ifa->ifa_addr->sa_family == AF_INET) {
            if (s != 0) {
                exit(EXIT_FAILURE);
            }
            printf("\tInterface : <%s>\n", ifa->ifa_name);
            printf("\t  Address : <%s>\n", host);
        }
    }

    freeifaddrs(ifaddr);
}

/*
 * setup ha_energy program to run as a background deamon
 * disconnect and exit foreground startup process
 */
void skeleton_daemon(void)
{
    pid_t pid;

    /* Fork off the parent process */
    pid = fork();

    /* An error occurred */
    if (pid < 0) {
        printf("\r\n%s DAEMON failure  LOG Version %s : MQTT Version %s\r\n", log_time(false),
LOG_VERSION, MQTT_VERSION);
        exit(EXIT_FAILURE);
    }

    /* Success: Let the parent terminate */
    if (pid > 0) {
        exit(EXIT_SUCCESS);
    }

    /* On success: The child process becomes session leader */
    if (setsid() < 0) {
        exit(EXIT_FAILURE);
    }

    /* Catch, ignore and handle signals */
    /*TODO: Implement a working signal handler */
    //    signal(SIGCHLD, SIG_IGN);
    //    signal(SIGHUP, SIG_IGN);

    /* Fork off for the second time*/
    pid = fork();

    /* An error occurred */
    if (pid < 0) {
        exit(EXIT_FAILURE);
    }

    /* Success: Let the parent terminate */
    if (pid > 0) {
        exit(EXIT_SUCCESS);
    }

    /* Set new file permissions */
    umask(0);

    /* Change the working directory to the root directory */
    /* or another appropriated directory */
    chdir("/");

    /* Close all open file descriptors */
    int x;
    for (x = sysconf(_SC_OPEN_MAX); x >= 0; x--) {
        close(x);
    }

}

/*
 * sent the current UTC to the Dump Load controller
 */
char * log_time(bool log)
{
    static char time_log[RBUF_SIZ] = {0};
    time_t rawtime_log;
    int32_t len = 0;

    tzset();
    timezone = 0;
    daylight = 0;
    time(&rawtime_log);
    sprintf(time_log, "%s", ctime(&rawtime_log));
    len = strlen(time_log);
    time_log[len - 1] = 0; // munge out the return character
    if (log) {
        fprintf(fout, "%s ", time_log);
        fflush(fout);
    }
    return time_log;
}

/*
 * data update timer flag
 * and 10 second software time clock
 */
void timer_callback(int32_t signum)
{
    signal(signum, timer_callback);
    ha_flag_vars_ss.runner = true;
    E.ten_sec_clock++;
    E.log_time_reset++;

}

/*
 * MQTT Broker connection errors can be fatal
 */
void connlost(void *context, char *cause)
{
    struct ha_flag_type *ha_flag = context;
    int32_t id_num = ha_flag->ha_id;
    static uint32_t times = 0;
    char * where = "Missing Topic";
    char * what = "Reconnection Error";

```

```c
    // bug-out if no context variables passed to callback
    if (context == NULL) {
        id_num = -1;
        goto bugout;
    }

    if (times++ > MQTT_RECONN) {
        goto bugout;
    } else {
        if (times > 1) {
            fprintf(fout, "%s Connection lost, retrying %d \n", log_time(false), times);
            fprintf(fout, "%s     cause: %s, h_id %d, c_id %d, %s \n", log_time(false), cause, id_num, 0, what);
            fprintf(fout, "%s MQTT DAEMON reconnect failure,  LOG Version %s : MQTT Version %s\n", log_time(false), LOG_VERSION, MQTT_VERSION);
        }
        fflush(fout);
        times = 0;
        return;
    }

bugout:
    fprintf(fout, "%s Connection lost, exit ha_energy program\n", log_time(false));
    fprintf(fout, "%s     cause: %s, h_id %d, c_id %d, %s \n", log_time(false), cause, id_num, 0, where);
    fprintf(fout, "%s MQTT DAEMON context is NULL failure,  LOG Version %s : MQTT Version %s\n", log_time(false), LOG_VERSION, MQTT_VERSION);
    fflush(fout);
    exit(EXIT_FAILURE);
}

/*
 * set the broker has message token
 */
void delivered(void *context, MQTTClient_deliveryToken dt)
{
    struct ha_flag_type *ha_flag = context;

    // bug-out if no context variables passed to callback
    if (context == NULL) {
        return;
    }
    ha_flag->deliveredtoken = dt;
}

void bmc_mqtt_init(void)
{
    E.mqtt_count = 0;
    gethostname(hname, hname_len);
    hname[12] = 0;
    printf("\r\n  LOG Version %s : MQTT Version %s : Host Name %s\r\n", LOG_VERSION, MQTT_VERSION, hname);
    showIP();
    skeleton_daemon();
#ifdef LOG_TO_FILE
    fout = fopen(LOG_TO_FILE, "a");
    if (fout == NULL) {
        fout = fopen(LOG_TO_FILE_ALT, "a");
        if (fout == NULL) {
            fout = stdout;
            printf("\r\n%s Unable to open LOG file %s \r\n", log_time(false), LOG_TO_FILE_ALT);
        }
    }
#else
    fout = stdout;
#endif

    /*
     * set the timer for MQTT publishing sample speed
     * CMD_SEC         10
     */
    setitimer(ITIMER_REAL, &new_timer, &old_timer);
    signal(SIGALRM, timer_callback);

    if (strncmp(hname, TNAME, 6) == 0) {
        MQTTClient_create(&E.client_p, LADDRESS, CLIENTID1,
            MQTTCLIENT_PERSISTENCE_NONE, NULL);
        conn_opts_p.keepAliveInterval = KAI;
        conn_opts_p.cleansession = 1;
        hname_ptr = LADDRESS;
    } else {
        MQTTClient_create(&E.client_p, ADDRESS, CLIENTID1,
            MQTTCLIENT_PERSISTENCE_NONE, NULL);
        conn_opts_p.keepAliveInterval = KAI;
        conn_opts_p.cleansession = 1;
        hname_ptr = ADDRESS;
    }

    fprintf(fout, "\r\n%s Connect MQTT server %s, %s\n", log_time(false), hname_ptr, CLIENTID1);
    fflush(fout);
    MQTTClient_setCallbacks(E.client_p, &ha_flag_vars_ss, connlost, msgarrvd, delivered);
    if ((E.rc = MQTTClient_connect(E.client_p, &conn_opts_p)) != MQTTCLIENT_SUCCESS) {
        fprintf(fout, "%s Failed to connect MQTT server, return code %d %s, %s\n", log_time(false), E.rc, hname_ptr, CLIENTID1);
        fflush(fout);
        pthread_mutex_destroy(&E.ha_lock);
        exit(EXIT_FAILURE);
    }

    MQTTClient_subscribe(E.client_p, TOPIC_P, QOS); // sub for testing data from the HA_Energy system

    pubmsg.payload = "online";
    pubmsg.payloadlen = strlen("online");
    pubmsg.qos = QOS;
    pubmsg.retained = 0;
    ha_flag_vars_ss.deliveredtoken = 0;
}

int32_t msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
{
    int32_t i, ret = 1;
    const char* payloadptr;
    char buffer[MBMQTT];
    struct ha_flag_type *ha_flag = context;

    E.mqtt_count++;
    // bug-out if no context variables passed to callback
    if (context == NULL) {
        ret = -1;
        goto null_exit;
    }

#ifdef DEBUG_REC
    fprintf(fout, "Message arrived\n");
#endif
    /*
     * move the received message into a processing holding buffer
     */
    payloadptr = message->payload;
    for (i = 0; i < message->payloadlen; i++) {
        buffer[i] = *payloadptr++;
    }
    buffer[i] = 0; // make a null terminated C string

    // parse the JSON data in the holding buffer
    cJSON *json = cJSON_ParseWithLength(buffer, message->payloadlen);
    if (json == NULL) {
        const char *error_ptr = cJSON_GetErrorPtr();
        if (error_ptr != NULL) {
            fprintf(fout, "%s Error: %s NULL cJSON pointer\n", log_time(false), error_ptr);
        }
        ret = -1;
        ha_flag->rec_ok = false;
        E.comedi = false;
        goto error_exit;
    }

    /*
     * MQTT messages for COMEDI
     */
#ifdef DEBUG_REC
    fprintf(fout, "COMEDI MQTT data\r\n");
#endif
    cJSON *data_result = json;

    data_result = cJSON_GetObjectItemCaseSensitive(json, "Comedi_Request");

    if (cJSON_IsString(data_result) && (data_result->valuestring != NULL)) {
        fprintf(fout, "%s Comedi Trigger from MQTT server, Topic %s %s\n", log_time(false), topicName, data_result->valuestring);
        fflush(fout);
        ret = true;
    }
    E.comedi = true;

    // done with processing MQTT async message, set state flags
    ha_flag->receivedtoken = true;
    ha_flag->rec_ok = true;
    ha_flag_vars_ss.runner = true; // send data in response to received message of any type
    /*
     * exit and delete/free resources. In steps depending of possible error conditions
     */
error_exit:
    // delete the JSON object
    cJSON_Delete(json);
null_exit:
    // free the MQTT objects
    MQTTClient_freeMessage(&message);
    MQTTClient_free(topicName);
    fflush(fout);
    return ret;
}

/*
 * send Comedi variables MQTT host
 */
void mqtt_bmc_data(MQTTClient client_p, const char * topic_p)
{
    cJSON *json;
    time_t rawtime;
    static uint32_t spam = 0;

    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    MQTTClient_deliveryToken token;
    ha_flag_vars_ss.deliveredtoken = 0;

    fprintf(fout, "%s Sending Comedi data to MQTT server, Topic %s\n", log_time(false), topic_p);
    fflush(fout);

    E.adc[0] = get_adc_volts(0);
    E.adc[1] = get_adc_volts(1);
    set_dac_volts(0, E.dac[0]);
    set_dac_volts(1, E.dac[1]);
    E.do_16b = bmc.dataout.dio_buf;
    E.di_16b = get_dio_bit(0)+ (get_dio_bit(1) « 1)+ (get_dio_bit(2) « 2)+ (get_dio_bit(3) « 3)+
(get_dio_bit(4) « 4);

    E.mqtt_count++;
    E.sequence++;
    json = cJSON_CreateObject();
    cJSON_AddStringToObject(json, "name", CLIENTID1);
    cJSON_AddNumberToObject(json, "sequence", E.sequence);
    cJSON_AddNumberToObject(json, "mqtt_do_16b", (double) E.do_16b);
    cJSON_AddNumberToObject(json, "http_di_16b", (double) E.di_16b);
    cJSON_AddNumberToObject(json, "bmc_adc0", E.adc[0]);
    cJSON_AddNumberToObject(json, "bmc_adc1", E.adc[1]);
    cJSON_AddNumberToObject(json, "bmc_dac0", E.dac[0]);
    cJSON_AddNumberToObject(json, "bmc_dac1", E.dac[1]);
    cJSON_AddStringToObject(json, "build_date", FW_Date);
    cJSON_AddStringToObject(json, "build_time", FW_Time);
    time(&rawtime);
    cJSON_AddNumberToObject(json, "sequence_time", (double) rawtime);
    // convert the cJSON object to a JSON string
    char *json_str = cJSON_Print(json);

    pubmsg.payload = json_str;
    pubmsg.payloadlen = strlen(json_str);
    pubmsg.qos = QOS;
    pubmsg.retained = 0;

    MQTTClient_publishMessage(client_p, topic_p, &pubmsg, &token);
    // a busy, wait loop for the async delivery thread to complete
    {
        uint32_t waiting = 0;
        while (ha_flag_vars_ss.deliveredtoken != token) {
            usleep(TOKEN_DELAY);
            if (waiting++ > MQTT_RETRY) {
                if (spam++ > 1) {
                    fprintf(fout, "%s SW mqtt_bmc_data, Still Waiting, timeout\r\n", log_time(false));
                    fflush(fout);
                    spam = 0;
                }
                break;
            } else {
                spam = 0;
            }
        };
    }

    cJSON_free(json_str);
    cJSON_Delete(json);
}

/*
 * main program function to send Comedi data to the MQTT server
 */
void comedi_push_mqtt(void)
{
    mqtt_bmc_data(E.client_p, TOPIC_P);
}
```

4.7 bmc_mqtt.h

```

#ifndef BMC_MQTT_H
#define BMC_MQTT_H

#ifdef __cplusplus
extern "C" {
#endif

#include "bmc.h"
#include "daq.h"
#include "mqtt_rec.h"
#include "mqtt_vars.h"

#define MQTT_RETRY 10

        extern struct ha_flag_type ha_flag_vars_ss;

        void mqtt_bmc_data(MQTTClient, const char *);
        void delivered(void *, MQTTClient_deliveryToken);
        int32_t msgarrvd(void *, char *, int, MQTTClient_message *);
        void connlost(void *, char *);
        void showIP(void);
        void skeleton_daemon(void);
        void bmc_mqtt_init(void);
        char * log_time(bool);

        void timer_callback(int32_t);
        void comedi_push_mqtt(void);

#ifdef __cplusplus
}
#endif

#endif /* BMC_MQTT_H */

```

4.8 bmc.o.d

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

4.9 bmc.o.d

```
build/Release/GNU-Linux/bmc.o: bmc.c daq.h bmc.h mqtt_rec.h mqtt_vars.h \
 bmc_mqtt.h
daq.h:
bmc.h:
mqtt_rec.h:
mqtt_vars.h:
bmc_mqtt.h:
```

4.10 daq.o.d

```
build/Debug/GNU-Linux/daq.o: daq.c daq.h
daq.h:
```

4.11 daq.o.d

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

4.12 bmc_mqtt.o.d

```
build/Release/GNU-Linux/bmc_mqtt.o: bmc_mqtt.c bmc_mqtt.h bmc.h daq.h \
 mqtt_rec.h mqtt_vars.h
bmc_mqtt.h:
bmc.h:
daq.h:
mqtt_rec.h:
mqtt_vars.h:
```

4.13 daq.c

```
/*
 * \file daq.c
 */



#include <stdio.h> /* for printf() */
#include <unistd.h>
#include <stdbool.h>
#include <stdint.h>
#include <comedilib.h>
#include "daq.h"

int subdev_ai = 0; /* change this to your input subdevice */
int chan_ai = 0; /* change this to your channel */
int range_ai = 0; /* more on this later */
int aref_ai = AREF_GROUND; /* more on this later */
int maxdata_ai, ranges_ai, channels_ai;

int subdev_ao = 0; /* change this to your input subdevice */
int chan_ao = 0; /* change this to your channel */
int range_ao = 0; /* more on this later */
int aref_ao = AREF_GROUND; /* more on this later */
int maxdata_ao, ranges_ao, channels_ao;

int subdev_di = 0; /* change this to your input subdevice */
int chan_di = 0; /* change this to your channel */
int range_di = 0; /* more on this later */
int maxdata_di, ranges_di, channels_di, datain_di;

int subdev_do = 0; /* change this to your input subdevice */
int chan_do = 0; /* change this to your channel */
int range_do = 0; /* more on this later */
int maxdata_do, ranges_do, channels_do, datain_do;

int subdev_dio; /* change this to your input subdevice */
int aref_dio; /* more on this later */

int subdev_counter; /* change this to your input subdevice */
int chan_counter = 0; /* change this to your channel */
int range_counter = 0; /* more on this later */
int maxdata_counter, ranges_counter, channels_counter, datain_counter;

comedi_t *it;
comedi_range *ad_range, *da_range;
bool ADC_OPEN = false, DIO_OPEN = false, ADC_ERROR = false, DEV_OPEN = false,
    DIO_ERROR = false, HAS_AO = false, DAC_ERROR = false, PWM_OPEN = false,
    PWM_ERROR = false;

int init_daq(double min_range, double max_range, int range_update)
{
    int i = 0;

    if (!DEV_OPEN) {
        it = comedi_open("/dev/comedi0");
        if (it == NULL) {
            comedi_perror("comedi_open");
            ADC_OPEN = false;
            DEV_OPEN = false;
            return -1;
        }
        DEV_OPEN = true;
    }

    subdev_ai = comedi_find_subdevice_by_type(it, COMEDI_SUBD_AI, subdev_ai);
    if (subdev_ai < 0) {
        return -2;
        ADC_OPEN = false;
    }


    subdev_ao = comedi_find_subdevice_by_type(it, COMEDI_SUBD_AO, subdev_ao);
    if (subdev_ao < 0) {
        HAS_AO = false;
    } else {
        HAS_AO = true;
    }

    fprintf(fout, "Comedi DAQ Board Name: %s, Driver: %s\r\n",
comedi_get_board_name(it),comedi_get_driver_name(it));

    fprintf(fout, "Subdev AI  %i ", subdev_ai);
    channels_ai = comedi_get_n_channels(it, subdev_ai);
    fprintf(fout, "Analog  Channels %i ", channels_ai);
    maxdata_ai = comedi_get_maxdata(it, subdev_ai, i);
    fprintf(fout, "Maxdata %i ", maxdata_ai);
    ranges_ai = comedi_get_n_ranges(it, subdev_ai, i);
    fprintf(fout, "Ranges %i ", ranges_ai);
    ad_range = comedi_get_range(it, subdev_ai, i, range_ai);
    if (range_update) {
        ad_range->min = min_range;
        ad_range->max = max_range;
    }
    fprintf(fout, ": ad_range .min = %.3f, max = %.3f\r\n", ad_range->min,
        ad_range->max);

    if (HAS_AO) {
        fprintf(fout, "Subdev AO  %i ", subdev_ao);
        channels_ao = comedi_get_n_channels(it, subdev_ao);
        fprintf(fout, "Analog  Channels %i ", channels_ao);
        maxdata_ao = comedi_get_maxdata(it, subdev_ao, i);
        fprintf(fout, "Maxdata %i ", maxdata_ao);
        ranges_ao = comedi_get_n_ranges(it, subdev_ao, i);
        fprintf(fout, "Ranges %i ", ranges_ao);
        da_range = comedi_get_range(it, subdev_ao, i, range_ao);
        fprintf(fout, ": da_range .min = %.3f, max = %.3f\r\n", da_range->min,
            da_range->max);
    }

    ADC_OPEN = true;
    comedi_set_global_oor_behavior(COMEDI_OOR_NUMBER);
    return 0;
}

int init_dac(double min_range, double max_range, int range_update)
{
    int i = 0;

    if (!DEV_OPEN) {
        it = comedi_open("/dev/comedi0");
        if (it == NULL) {
            comedi_perror("comedi_open");
            ADC_OPEN = false;
            DEV_OPEN = false;
            return -1;
        }
        DEV_OPEN = true;
    }

    subdev_ao = comedi_find_subdevice_by_type(it, COMEDI_SUBD_AO, subdev_ao);
    if (subdev_ao < 0) {
        HAS_AO = false;
    } else {
        HAS_AO = true;
    }

    if (HAS_AO) {
        fprintf(fout, "Subdev AO  %i ", subdev_ao);
        channels_ao = comedi_get_n_channels(it, subdev_ao);
        fprintf(fout, "Analog  Channels %i ", channels_ao);
        maxdata_ao = comedi_get_maxdata(it, subdev_ao, i);
        fprintf(fout, "Maxdata %i ", maxdata_ao);
        ranges_ao = comedi_get_n_ranges(it, subdev_ao, i);
        fprintf(fout, "Ranges %i ", ranges_ao);
        da_range = comedi_get_range(it, subdev_ao, i, range_ao);
        fprintf(fout, ": da_range .min = %.3f, max = %.3f\r\n", da_range->min,
            da_range->max);
    }

    comedi_set_global_oor_behavior(COMEDI_OOR_NUMBER);
    return 0;
}

int adc_range(double min_range, double max_range)
{
    if (ADC_OPEN) {
        ad_range->min = min_range;
        ad_range->max = max_range;
        return 0;
    } else {
        return -1;
    }
}

int dac_range(double min_range, double max_range)
{
    if (ADC_OPEN) {
        da_range->min = min_range;
        da_range->max = max_range;
        return 0;
    } else {
        return -1;
    }
}

int set_dac_volts(int chan, double voltage)
{
    lsampl_t data;
    int retval;

    data = comedi_from_phys(voltage, da_range, maxdata_ao);
    bmc.dac_sample[chan] = data;
    retval = comedi_data_write(it, subdev_ao, chan, range_ao, aref_ao, data);
    if (retval < 0) {
        comedi_perror("comedi_data_write in set_dac_volts");
        DAC_ERROR = true;
    }
    return retval;
}

int set_dac_raw(int chan, lsampl_t voltage)
{
    int retval;

    retval = comedi_data_write(it, subdev_ao, chan, range_ao, aref_ao, voltage);
    if (retval < 0) {
        comedi_perror("comedi_data_write in set_dac_raw");
        DAC_ERROR = true;
    }
    return retval;
}

double get_adc_volts(int chan)
{
    lsampl_t data[16];
    int retval;

    retval = comedi_data_read_n(it, subdev_ai, chan, range_ai, aref_ai, &data[0], 8);
    if (retval < 0) {
        comedi_perror("comedi_data_read in get_adc_volts");
        ADC_ERROR = true;
        return 0.0;
    }
    bmc.adc_sample[chan] = data[0];
    return comedi_to_phys(data[0], ad_range, maxdata_ai);
}

int set_dio_output(int chan)
{
    return comedi_dio_config(it,
        subdev_dio,
        chan,
        COMEDI_OUTPUT);
}

int set_dio_input(int chan)
{
    return comedi_dio_config(it,
        subdev_dio,
        chan,
        COMEDI_INPUT);
}

int get_dio_bit(int chan)
{
    lsampl_t data;
    int retval;

    retval = comedi_data_read(it, subdev_di, chan, range_di, aref_dio, &data);
    if (retval < 0) {
        comedi_perror("comedi_data_read in get_dio_bits");
        DIO_ERROR = true;
        return 0;
    }
    return data;
}

int put_dio_bit(int chan, int bit_data)
{
    lsampl_t data = bit_data;
    int retval;

    retval = comedi_data_write(it, subdev_do, chan, range_do, aref_dio, data);
    if (retval < 0) {
        comedi_perror("comedi_data_write in put_dio_bits");
        DIO_ERROR = true;
        return -1;
    }
    return 0;
}

int init_dio(void)
{
    int i = 0;

    if (!DEV_OPEN) {
        it = comedi_open("/dev/comedi0");
        if (it == NULL) {
            comedi_perror("comedi_open");
            DIO_OPEN = false;
            DEV_OPEN = false;
            return -1;
        }
        DEV_OPEN = true;
    }

    subdev_di = comedi_find_subdevice_by_type(it, COMEDI_SUBD_DI, subdev_di);
    if (subdev_di < 0) {
        return -1;
        DIO_OPEN = false;
    }
    subdev_do = comedi_find_subdevice_by_type(it, COMEDI_SUBD_DO, subdev_do);
    if (subdev_do < 0) {
        return -1;
        DIO_OPEN = false;
    }

    subdev_counter = comedi_find_subdevice_by_type(it, COMEDI_SUBD_COUNTER, subdev_counter);
    if (subdev_counter < 0) {
        return -1;
        PWM_OPEN = false;
    }

    fprintf(fout, "Subdev DI  %i ", subdev_di);
    channels_di = comedi_get_n_channels(it, subdev_di);
    fprintf(fout, "Digital Channels %i ", channels_di);
    maxdata_di = comedi_get_maxdata(it, subdev_di, i);
    fprintf(fout, "Maxdata %i ", maxdata_di);
    ranges_di = comedi_get_n_ranges(it, subdev_di, i);
    fprintf(fout, "Ranges %i \r\n", ranges_di);

    fprintf(fout, "Subdev DO  %i ", subdev_do);
    channels_do = comedi_get_n_channels(it, subdev_do);
    fprintf(fout, "Digital Channels %i ", channels_do);
    maxdata_do = comedi_get_maxdata(it, subdev_do, i);
    fprintf(fout, "Maxdata %i ", maxdata_do);
    ranges_do = comedi_get_n_ranges(it, subdev_do, i);
    fprintf(fout, "Ranges %i \r\n", ranges_do);

    fprintf(fout, "Subdev COU %i ", subdev_counter);
    channels_counter = comedi_get_n_channels(it, subdev_counter);
    fprintf(fout, "Digital Channels %i ", channels_counter);
    maxdata_counter = comedi_get_maxdata(it, subdev_counter, i);
    fprintf(fout, "Maxdata %i ", maxdata_counter);
    ranges_counter = comedi_get_n_ranges(it, subdev_counter, i);
    fprintf(fout, "Ranges %i \r\n", ranges_counter);
    DIO_OPEN = true;
    return 0;
}

int get_data_sample(void)
{
    unsigned int obits;

    bmc.datain.D0 = get_dio_bit(0);

    if (JUST_BITS) { // send I/O bit by bit
        put_dio_bit(0, bmc.dataout.d.D0);
        put_dio_bit(1, bmc.dataout.d.D1);
        put_dio_bit(2, bmc.dataout.d.D2);
        put_dio_bit(3, bmc.dataout.d.D3);
        put_dio_bit(4, bmc.dataout.d.D4);
        put_dio_bit(5, bmc.dataout.d.D5);
        put_dio_bit(6, bmc.dataout.d.D6);
        put_dio_bit(7, bmc.dataout.d.D7);
    } else { // send I/O as a byte mask
        obits = bmc.dataout.dio_buf;
        comedi_dio_bitfield2(it, subdev_do, 0xff, &obits, 0);
    }

    return 0;
}

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
{
    static double smooth[LPCHANC] = {0};
    double lp_speed, lp_x;

    if ((bn >= LPCHANC) || (bn < 0)) // check for proper array position
        return new;
    if (slow) {
        lp_speed = 0.033;
    } else {
        lp_speed = 0.125;
    }
    lp_x = ((smooth[bn]*100.0) + (((new * 100.0)-(smooth[bn]*100.0)) * lp_speed)) / 100.0;
    smooth[bn] = lp_x;
    if (slow == (-1)) { // reset and return zero
        lp_x = 0.0;
        smooth[bn] = 0.0;
    }
    return lp_x;
}
```

4.14 daq.h

```
/*
 * File:   daq.h
 * Author: root
 *
 * Created on September 21, 2012, 6:49 PM
 */

#ifndef DAQ_H
#define DAQ_H

#ifdef __cplusplus
extern "C" {
#endif

#define PVV_C   0
#define CCV_C   1
#define SYV_C   2
#define B1V_C   3
#define B2V_C   4
#define INV_C   5
#define VD5_C   7
#define PVC_C   8
#define CCC_C   9
#define BAC_C   10

#define LPCHANC        16

#define JUST_BITS false

#include <stdint.h>
#include <comedilib.h>
#include "bmc.h"

        struct didata {
                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; //
        };

        union dio_buf_type {
                uint32_t dio_buf;
                struct didata d;
        };

        typedef struct bmcdata {
                double pv_voltage, cc_voltage, input_voltage, b1_voltage, b2_voltage, system_voltage, logic_voltage;
                double pv_current, cc_current, battery_current;
                struct didata datain;
                union dio_buf_type dataout;
                int32_t adc_sample[32];
                int32_t dac_sample[32];
                int32_t utc;
        }
        bmctype;

        extern volatile struct bmcdata bmc;
        extern struct didata datain;
        extern struct dodata dataout;

        extern int maxdata_ai, ranges_ai, channels_ai;
        extern int maxdata_ao, ranges_ao, channels_ao;
        extern int maxdata_di, ranges_di, channels_di, datain_di;
        extern int maxdata_do, ranges_do, channels_do, datain_do;
        extern int maxdata_counter, ranges_counter, channels_counter, datain_counter;

        int init_daq(double, double, int);
        int init_dac(double, double, int);
        int init_dio(void);
        int adc_range(double, double);
        int dac_range(double, double);
        double get_adc_volts(int);
        int set_dac_volts(int, double);
        int set_dac_raw(int, lsampl_t);
        int get_dio_bit(int);
        int put_dio_bit(int, int);
        int set_dio_input(int);
        int set_dio_output(int);
        int get_data_sample(void);
        double lp_filter(double, int, int);
#ifdef __cplusplus
}
#endif

#endif /* DAQ_H */

```

4.15 mqtt_rec.h

```


#ifndef MQTT_REC_H
#define MQTT_REC_H

#ifdef __cplusplus
extern "C" {
#endif

#include "mqtt_vars.h"

#define RDEV_SIZE        10

#define SLEEP_CODE      0
#define FLOAT_CODE      1
        //#define DEBUG_REC
        //#define GET_DEBUG

#define MBMQTT  1024

        enum mqtt_id {
                P8055_ID,
                FM80_ID,
                DUMPLOAD_ID,
                HA_ID,
                COMEDI_ID,
                LAST_MQTT_ID,
        };

        struct ha_flag_type {
                volatile MQTTClient_deliveryToken deliveredtoken, receivedtoken;
                volatile bool runner, rec_ok;
                int32_t ha_id;
                volatile int32_t var_update, energy_mode;
        };

        extern FILE* fout;

        int32_t msgarrvd(void *, char *, int, MQTTClient_message *);
        void delivered(void *, MQTTClient_deliveryToken);

        bool json_get_data(cJSON *, const char *, cJSON *, uint32_t);
        bool fm80_float(const bool set_bias);
        bool fm80_sleep(void);


#ifdef __cplusplus
}
#endif

#endif /* MQTT_REC_H */

```

4.16 mqtt_vars.h

```


#ifndef MQTT_VARS_H
#define MQTT_VARS_H

#ifdef __cplusplus
extern "C" {
#endif

#define HA_SW_DELAY     400000  // usecs
#define TOKEN_DELAY     600
#define GTI_TOKEN_DELAY 300

#define QOS             1

        void mqtt_ha_switch(MQTTClient, const char *, const bool);
        void mqtt_ha_pid(MQTTClient, const char *);
        void mqtt_ha_shutdown(MQTTClient, const char *);
        bool mqtt_gti_power(MQTTClient, const char *, char *, uint32_t);
        bool mqtt_gti_time(MQTTClient, const char *, char *);


#ifdef __cplusplus
}
#endif

#endif /* MQTT_VARS_H */

```

4.17 c_standard_headers_indexer.c

```
/*
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS HEADER.
 *
 * Copyright (c) 2016 Oracle and/or its affiliates. All rights reserved.
 *
 * Oracle and Java are registered trademarks of Oracle and/or its affiliates.
 * Other names may be trademarks of their respective owners.
 *
 * The contents of this file are subject to the terms of either the GNU
 * General Public License Version 2 only ("GPL") or the Common
 * Development and Distribution License("CDDL") (collectively, the
 * "License"). You may not use this file except in compliance with the
 * License. You can obtain a copy of the License at
 * http://www.netbeans.org/cddl-gplv2.html
 * or nbbuild/licenses/CDDL-GPL-2-CP. See the License for the
 * specific language governing permissions and limitations under the
 * License.  When distributing the software, include this License Header
 * Notice in each file and include the License file at
 * nbbuild/licenses/CDDL-GPL-2-CP.  Oracle designates this
 * particular file as subject to the "Classpath" exception as provided
 * by Oracle in the GPL Version 2 section of the License file that
 * accompanied this code. If applicable, add the following below the
 * License Header, with the fields enclosed by brackets [] replaced by
 * your own identifying information:
 * "Portions Copyrighted [year] [name of copyright owner]"
 *
 * If you wish your version of this file to be governed by only the CDDL
```

```

// List of standard headers was taken in http://en.cppreference.com/w/c/header

#include <assert.h>      // Conditionally compiled macro that compares its argument to zero
#include <ctype.h>   // Functions to determine the type contained in character data
#include <errno.h>   // Macros reporting error conditions
#include <float.h>   // Limits of float types
#include <limits.h>      // Sizes of basic types
#include <locale.h>      // Localization utilities
#include <math.h>    // Common mathematics functions
#include <setjmp.h>      // Nonlocal jumps
#include <signal.h>      // Signal handling
#include <stdarg.h>      // Variable arguments
#include <stddef.h>      // Common macro definitions
#include <stdio.h>   // Input/output
#include <string.h>      // String handling
#include <stdlib.h>      // General utilities: memory management, program utilities, string
conversions, random numbers
#include <time.h>    // Time/date utilities
#include <iso646.h>      // (since C95) Alternative operator spellings
#include <wchar.h>       // (since C95) Extended multibyte and wide character utilities
#include <wctype.h>      // (since C95) Wide character classification and mapping utilities
#ifdef _STDC_C99
#include <complex.h>     // (since C99) Complex number arithmetic
#include <fenv.h>        // (since C99) Floating-point environment
#include <inttypes.h>    // (since C99) Format conversion of integer types
#include <stdbool.h>     // (since C99) Boolean type
#include <stdint.h>      // (since C99) Fixed-width integer types
#include <tgmath.h>      // (since C99) Type-generic math (macros wrapping math.h and complex.h)
#endif
#ifdef _STDC_C11
#include <stdalign.h>    // (since C11) alignas and alignof convenience macros
#include <stdatomic.h>   // (since C11) Atomic types
#include <stdnoreturn.h> // (since C11) noreturn convenience macros
#include <threads.h>     // (since C11) Thread library
#include <uchar.h>       // (since C11) UTF-16 and UTF-32 character utilities
#endif
```

4.18 cpp_standard_headers_indexer.cpp

```
 * your version of this file under either the CDDL, the GPL Version 2 or
 * to extend the choice of license to its licensees as provided above.
 * However, if you add GPL Version 2 code and therefore, elected the GPL
 * Version 2 license, then the option applies only if the new code is
 * made subject to such option by the copyright holder.
 *
 * Contributor(s):
 */

// List of standard headers was taken in http://en.cppreference.com/w/cpp/header

#include <cstdlib>      // General purpose utilities: program control, dynamic memory allocation, random numbers, sort and search
#include <csignal>      // Functions and macro constants for signal management
#include <csetjmp>      // Macro (and function) that saves (and jumps) to an execution context
#include <cstdarg>      // Handling of variable length argument lists
#include <typeinfo>         // Runtime type information utilities
#include <bitset>       // std::bitset class template
#include <functional>       // Function objects, designed for use with the standard algorithms
#include <utility>      // Various utility components
#include <ctime>        // C-style time/date utilites
#include <cstddef>      // typedefs for types such as size_t, NULL and others
#include <new>              // Low-level memory management utilities
#include <memory>       // Higher level memory management utilities
#include <climits>          // limits of integral types
#include <cfloat>       // limits of float types
#include <limits>       // standardized way to query properties of arithmetic types
#include <exception>        // Exception handling utilities
#include <stdexcept>        // Standard exception objects
#include <cassert>      // Conditionally compiled macro that compares its argument to zero
#include <cerrno>           // Macro containing the last error number
#include <cctype>           // functions to determine the type contained in character data
#include <cwctype>          // functions for determining the type of wide character data
#include <cstring>      // various narrow character string handling functions
#include <cwchar>       // various wide and multibyte string handling functions
#include <string>       // std::basic_string class template
#include <vector>       // std::vector container
#include <deque>        // std::deque container
#include <list>         // std::list container
#include <set>              // std::set and std::multiset associative containers
#include <map>              // std::map and std::multimap associative containers
#include <stack>        // std::stack container adaptor
#include <queue>        // std::queue and std::priority_queue container adaptors
#include <algorithm>        // Algorithms that operate on containers
#include <iterator>         // Container iterators
#include <cmath>            // Common mathematics functions
#include <complex>          // Complex number type
#include <valarray>         // Class for representing and manipulating arrays of values
#include <numeric>          // Numeric operations on values in containers
#include <iosfwd>           // forward declarations of all classes in the input/output library
#include <ios>              // std::ios_base class, std::basic_ios class template and several typedefs
#include <istream>          // std::basic_istream class template and several typedefs
#include <ostream>          // std::basic_ostream, std::basic_iostream class templates and several typedefs
#include <iostream>         // several standard stream objects
#include <fstream>          // std::basic_fstream, std::basic_ifstream, std::basic_ofstream class templates and several typedefs
#include <sstream>          // std::basic_stringstream, std::basic_istringstream, std::basic_ostringstream class templates and several typedefs
#include <strstream>        // std::strstream, std::istrstream, std::ostrstream(deprecated)
#include <iomanip>          // Helper functions to control the format or input and output
#include <streambuf>        // std::basic_streambuf class template
#include <cstdio>           // C-style input-output functions
#include <locale>           // Localization utilities
#include <clocale>          // C localization utilities
#include <ciso646>          // empty header. The macros that appear in iso646.h in C are keywords in C++
#if __cplusplus >= 201103L
#include <typeindex>        // (since C++11)    std::type_index
#include <type_traits>      // (since C++11)    Compile-time type information
#include <chrono>           // (since C++11)    C++ time utilites
#include <initializer_list> // (since C++11)    std::initializer_list class template
#include <tuple>            // (since C++11)    std::tuple class template
#include <scoped_allocator> // (since C++11)    Nested allocator class
#include <cstdint>          // (since C++11)    fixed-size types and limits of other types
#include <cinttypes>        // (since C++11)    formatting macros , intmax_t and uintmax_t math and conversions
#include <system_error>     // (since C++11)    defines std::error_code, a platform-dependent error code
#include <cuchar>           // (since C++11)    C-style Unicode character conversion functions
#include <array>            // (since C++11)    std::array container
#include <forward_list>     // (since C++11)    std::forward_list container
#include <unordered_set>    // (since C++11)    std::unordered_set and std::unordered_multiset unordered associative containers
#include <unordered_map>    // (since C++11)    std::unordered_map and std::unordered_multimap unordered associative containers
#include <random>           // (since C++11)    Random number generators and distributions
```

```
#include <ratio>            // (since C++11)    Compile-time rational arithmetic
#include <cfenv>            // (since C++11)    Floating-point environment access functions
#include <codecvt>          // (since C++11)    Unicode conversion facilities
#include <regex>            // (since C++11)    Classes, algorithms and iterators to support regular expression processing
#include <atomic>           // (since C++11)    Atomic operations library
#include <ccomplex>         // (since C++11)(deprecated in C++17)   simply includes the header <complex>
#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
#include <cstdalign>        // (since C++11)(deprecated in C++17)   defines one compatibility macro constant
#include <cstdbool>         // (since C++11)(deprecated in C++17)   defines one compatibility macro constant
#include <thread>           // (since C++11)    std::thread class and supporting functions
#include <mutex>            // (since C++11)    mutual exclusion primitives
#include <future>           // (since C++11)    primitives for asynchronous computations
#include <condition_variable> // (since C++11)  thread waiting conditions
#endif
#if __cplusplus >= 201300L
#include <shared_mutex>     // (since C++14)    shared mutual exclusion primitives
#endif
#if __cplusplus >= 201500L
#include <any>              // (since C++17)    std::any class template
#include <optional>         // (since C++17)    std::optional class template
#include <variant>          // (since C++17)    std::variant class template
#include <memory_resource>  // (since C++17)    Polymorphic allocators and memory resources
#include <string_view>      // (since C++17)    std::basic_string_view class template
#include <execution>        // (since C++17)    Predefined execution policies for parallel versions of the algorithms
#include <filesystem>       // (since C++17)    std::path class and supporting functions
#endif
```

Index