My Project

1.0

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

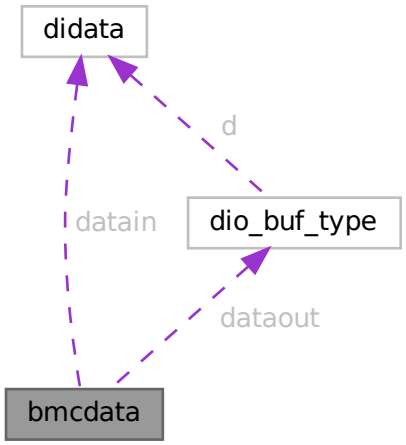
Here is a list of all documented files with brief descriptions:

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
- board_t BOARD
- char * BNAME

3.1.1 Detailed Description

Definition at line 88 of file daq.h.

3.1.2 Field Documentation

3.1.2.1 adc_sample

```
int32_t adc_sample[32]
```

Definition at line 93 of file daq.h.

3.1.2.2 b1_voltage

```
double b1_voltage
```

Definition at line 89 of file daq.h.

3.1.2.3 b2_voltage

```
double b2_voltage
```

Definition at line 89 of file daq.h.

3.1.2.4 battery_current

```
double battery_current
```

Definition at line 90 of file daq.h.

3.1.2.5 BNAME

```
char* BNAME
```

Definition at line 97 of file daq.h.

3.1.2.6 BOARD

```
board_t BOARD
```

Definition at line 96 of file daq.h.

3.1.2.7 cc_current

```
double cc_current
```

Definition at line 90 of file daq.h.

3.1.2.8 cc_voltage

```
double cc_voltage
```

Definition at line 89 of file daq.h.

3.1.2.9 dac_sample

```
int32_t dac_sample[32]
```

Definition at line 94 of file daq.h.

3.1.2.10 datain

```
struct didata datain
```

Definition at line 91 of file daq.h.

3.1.2.11 dataout

```
union dio_buf_type dataout
```

Definition at line 92 of file daq.h.

3.1.2.12 input_voltage

```
double input_voltage
```

Definition at line 89 of file daq.h.

3.1.2.13 logic_voltage

```
double logic_voltage
```

Definition at line 89 of file daq.h.

3.1.2.14 pv_current

```
double pv_current
```

Definition at line 90 of file daq.h.

3.1.2.15 pv_voltage

```
double pv_voltage
```

Definition at line 89 of file daq.h.

3.1.2.16 system_voltage

```
double system_voltage
```

Definition at line 89 of file daq.h.

3.1.2.17 utc

```
int32_t utc
```

Definition at line 95 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 72 of file daq.h.

3.2.2 Field Documentation

3.2.2.1 D0

```
uint32_t D0
```

Definition at line 73 of file daq.h.

3.2.2.2 D1

```
uint32_t D1
```

Definition at line 74 of file daq.h.

3.2.2.3 D2

```
uint32_t D2
```

Definition at line 75 of file daq.h.

3.2.2.4 D3

```
uint32_t D3
```

Definition at line 76 of file daq.h.

3.2.2.5 D4

```
uint32_t D4
```

Definition at line 77 of file daq.h.

3.2.2.6 D5

```
uint32_t D5
```

Definition at line 78 of file daq.h.

3.2.2.7 D6

```
uint32_t D6
```

Definition at line 79 of file daq.h.

3.2.2.8 D7

```
uint32_t D7
```

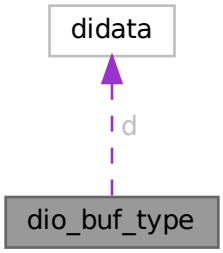
Definition at line 80 of file daq.h.

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 83 of file daq.h.

3.3.2 Field Documentation

3.3.2.1 d

```
struct didata d
```

Definition at line 85 of file daq.h.

3.3.2.2 dio_buf

```
uint32_t dio_buf
```

Definition at line 84 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 [ANA_BUFFERS]
- double dac [ANA_BUFFERS]
- MQTTClient client_p
- MQTTClient client_sd
- MQTTClient client_ha

3.4.1 Detailed Description

Definition at line 92 of file bmc.h.

3.4.2 Field Documentation

3.4.2.1 ac_low_adj

```
volatile double ac_low_adj
```

Definition at line 94 of file bmc.h.

3.4.2.2 ac_mismatch

```
volatile bool ac_mismatch
```

Definition at line 95 of file bmc.h.

3.4.2.3 ac_sw_on

```
volatile bool ac_sw_on
```

Definition at line 95 of file bmc.h.

3.4.2.4 ac_sw_status

```
volatile bool ac_sw_status
```

Definition at line 95 of file bmc.h.

3.4.2.5 adc

```
double adc[ANA_BUFFERS]
```

Definition at line 101 of file bmc.h.

3.4.2.6 client_ha

```
MQTTClient client_ha
```

Definition at line 102 of file bmc.h.

3.4.2.7 client_p

```
MQTTClient client_p
```

Definition at line 102 of file bmc.h.

3.4.2.8 client_sd

```
MQTTClient client_sd
```

Definition at line 102 of file bmc.h.

3.4.2.9 comedi

```
volatile bool comedi
```

Definition at line 93 of file bmc.h.

3.4.2.10 dac

```
double dac[ANA_BUFFERS]
```

Definition at line 101 of file bmc.h.

3.4.2.11 dc_mismatch

```
volatile bool dc_mismatch
```

Definition at line 95 of file bmc.h.

3.4.2.12 di_16b

```
volatile int16_t di_16b
```

Definition at line 100 of file bmc.h.

3.4.2.13 dl_excess

```
volatile bool dl_excess
```

Definition at line 95 of file bmc.h.

3.4.2.14 dl_excess_adj

```
volatile double dl_excess_adj
```

Definition at line 94 of file bmc.h.

3.4.2.15 do_16b

```
volatile int16_t do_16b
```

Definition at line 100 of file bmc.h.

3.4.2.16 dumpload

```
volatile bool dumpload
```

Definition at line 93 of file bmc.h.

3.4.2.17 fm80

```
volatile bool fm80
```

Definition at line 93 of file bmc.h.

3.4.2.18 gti_delay

```
volatile uint32_t gti_delay
```

Definition at line 96 of file bmc.h.

3.4.2.19 gti_low_adj

```
volatile double gti_low_adj
```

Definition at line 94 of file bmc.h.

3.4.2.20 gti_sw_on

```
volatile bool gti_sw_on
```

Definition at line 95 of file bmc.h.

3.4.2.21 gti_sw_status

```
volatile bool gti_sw_status
```

Definition at line 95 of file bmc.h.

3.4.2.22 ha_lock

```
pthread_mutex_t ha_lock
```

Definition at line 99 of file bmc.h.

3.4.2.23 homeassistant

```
volatile bool homeassistant
```

Definition at line 93 of file bmc.h.

3.4.2.24 iammeter

```
volatile bool iammeter
```

Definition at line 93 of file bmc.h.

3.4.2.25 im_delay

```
volatile uint32_t im_delay
```

Definition at line 96 of file bmc.h.

3.4.2.26 im_display

```
volatile uint32_t im_display
```

Definition at line 96 of file bmc.h.

3.4.2.27 log_spam

```
volatile uint32_t log_spam
```

Definition at line 98 of file bmc.h.

3.4.2.28 log_time_reset

```
volatile uint32_t log_time_reset
```

Definition at line 98 of file bmc.h.

3.4.2.29 mode_mismatch

```
volatile bool mode_mismatch
```

Definition at line 95 of file bmc.h.

3.4.2.30 mqtt_count

```
volatile uint32_t mqtt_count
```

Definition at line 96 of file bmc.h.

3.4.2.31 once_ac

```
volatile bool once_ac
```

Definition at line 93 of file bmc.h.

3.4.2.32 once_gti

```
volatile bool once_gti
```

Definition at line 93 of file bmc.h.

3.4.2.33 once_gti_zero

```
volatile bool once_gti_zero
```

Definition at line 93 of file bmc.h.

3.4.2.34 rc

```
volatile int32_t rc
```

Definition at line 97 of file bmc.h.

3.4.2.35 sane

```
volatile int32_t sane
```

Definition at line 97 of file bmc.h.

3.4.2.36 sequence

```
volatile uint32_t sequence
```

Definition at line 96 of file bmc.h.

3.4.2.37 solar_mode

```
volatile bool solar_mode
```

Definition at line 95 of file bmc.h.

3.4.2.38 solar_shutdown

```
volatile bool solar_shutdown
```

Definition at line 95 of file bmc.h.

3.4.2.39 speed_go

```
volatile uint32_t speed_go
```

Definition at line 96 of file bmc.h.

3.4.2.40 startup

```
volatile bool startup
```

Definition at line 95 of file bmc.h.

3.4.2.41 thirty_sec_clock

```
volatile uint32_t thirty_sec_clock
```

Definition at line 98 of file bmc.h.

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

- bmc.h

3.5 ha_daq_hosts_type Struct Reference

Data Fields

- const char hosts [4][NI_MAXHOST]
- const char clients [4][NI_MAXHOST]
- const char topics [4][NI_MAXHOST]
- char hname [4][NI_MAXHOST]
- double scaler [4]
- uint8_t hindex

3.5.1 Detailed Description

Definition at line 45 of file bmc_mqtt.h.

3.5.2 Field Documentation

3.5.2.1 clients

```
const char clients[4][NI_MAXHOST]
```

Definition at line 47 of file bmc_mqtt.h.

3.5.2.2 hindex

```
uint8_t hindex
```

Definition at line 51 of file bmc_mqtt.h.

3.5.2.3 hname

```
char hname[4][NI_MAXHOST]
```

Definition at line 49 of file bmc_mqtt.h.

3.5.2.4 hosts

```
const char hosts[4][NI_MAXHOST]
```

Definition at line 46 of file bmc_mqtt.h.

3.5.2.5 scaler

```
double scaler[4]
```

Definition at line 50 of file bmc_mqtt.h.

3.5.2.6 topics

```
const char topics[4][NI_MAXHOST]
```

Definition at line 48 of file bmc_mqtt.h.

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

- bmc_mqtt.h

3.6 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

3.6.1 Detailed Description

Definition at line 38 of file bmc_mqtt.h.

3.6.2 Field Documentation

3.6.2.1 deliveredtoken

```
volatile MQTTClient_deliveryToken deliveredtoken
```

Definition at line 39 of file bmc_mqtt.h.

3.6.2.2 energy_mode

```
volatile int32_t energy_mode
```

Definition at line 42 of file bmc_mqtt.h.

3.6.2.3 ha_id

```
int32_t ha_id
```

Definition at line 41 of file bmc_mqtt.h.

3.6.2.4 rec_ok

```
volatile bool rec_ok
```

Definition at line 40 of file bmc_mqtt.h.

3.6.2.5 receivedtoken

```
volatile MQTTClient_deliveryToken receivedtoken
```

Definition at line 39 of file bmc_mqtt.h.

3.6.2.6 runner

```
volatile bool runner
```

Definition at line 40 of file bmc_mqtt.h.

3.6.2.7 var_update

```
volatile int32_t var_update
```

Definition at line 42 of file bmc_mqtt.h.

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

- bmc_mqtt.h

3.7 mcp2210_spi_type Struct Reference

Data Fields

- hid_device * handle
- struct hid_device_info * devs
- struct hid_device_info * cur_dev
- uint8_t buf [COMMAND_BUFFER_LENGTH]
- uint8_t rbuf [RESPONSE_BUFFER_LENGTH]
- int32_t res

3.7.1 Detailed Description

Definition at line 53 of file bmcdio.h.

3.7.2 Field Documentation

3.7.2.1 buf

```
uint8_t buf[COMMAND_BUFFER_LENGTH]
```

Definition at line 56 of file bmcdio.h.

3.7.2.2 cur_dev

```
struct hid_device_info * cur_dev
```

Definition at line 55 of file bmcdio.h.

3.7.2.3 devs

```
struct hid_device_info* devs
```

Definition at line 55 of file bmcdio.h.

3.7.2.4 handle

```
hid_device* handle
```

Definition at line 54 of file bmcdio.h.

3.7.2.5 rbuf

```
uint8_t rbuf[RESPONSE_BUFFER_LENGTH]
```

Definition at line 57 of file bmcdio.h.

3.7.2.6 res

```
int32_t res
```

Definition at line 58 of file bmcdio.h.

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

- bmcdio.h

Chapter 4

File Documentation

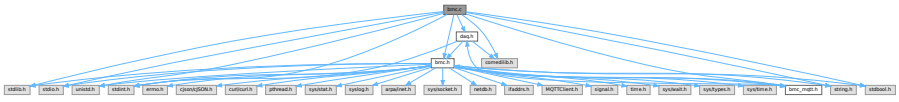
4.1 .dep.inc

```
00001 # This code depends on make tool being used
00002 DEPFILES=$(wildcard $(addsuffix .d, ${OBJECTFILES} ${TESTOBJECTFILES}))
00003 ifneq (${DEPFILES},)
00004 include ${DEPFILES}
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 "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 97 of file bmc.c.

4.2.2.2 main()

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

Definition at line 131 of file bmc.c.

4.2.3 Variable Documentation

4.2.3.1 bmc

```
volatile struct bmcdata bmc
```

Initial value:

```
= {
    .BOARD = bmcboard,
    .BNAME = "BMCBoard",
}
```

Definition at line 20 of file bmc.c.

4.2.3.2 board_name

```
const char* board_name = "NO_BOARD"
```

Definition at line 56 of file bmc.c.

4.2.3.3 driver_name

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

Definition at line 56 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 25 of file bmc.c.

4.2.3.5 fout

```
FILE* fout
```

Definition at line 58 of file bmc.c.

4.2.3.6 sine_wave

```
uint8_t sine_wave[256]
```

Definition at line 62 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 "bmc_mqtt.h"
00019
00020 volatile struct bmcdata bmc = {
```

```
00021     .BOARD = bmcboard,
00022     .BNAME = "BMCBoard",
00023 }; /* DAQ buffer */
00024
00025 struct energy_type E = {
00026     .once_gti = true,
00027     .once_ac = true,
00028     .once_gti_zero = true,
00029     .iammeter = false,
00030     .fm80 = false,
00031     .dumpload = false,
00032     .homeassistant = false,
00033     .ac_low_adj = 0.0f,
00034     .gti_low_adj = 0.0f,
00035     .ac_sw_on = true,
00036     .gti_sw_on = true,
00037     .im_delay = 0,
00038     .gti_delay = 0,
00039     .im_display = 0,
00040     .rc = 0,
00041     .speed_go = 0,
00042     .ac_sw_status = false,
00043     .gti_sw_status = false,
00044     .solar_mode = false,
00045     .solar_shutdown = false,
00046     .startup = true,
00047     .ac_mismatch = false,
00048     .dc_mismatch = false,
00049     .mode_mismatch = false,
00050     .dl_excess = false,
00051     .dl_excess_adj = 0.0f,
00052 };
00053
00054
00055 // Comedi I/O device type
00056 const char *board_name = "NO_BOARD", *driver_name = "NO_DRIVER";
00057
00058 FILE* fout; // logging stream
00059
00060 /* ripped from http://aquaticus.info/pwm-sine-wave */
00061
00062 uint8_t sine_wave[256] = {
00063     0x80, 0x83, 0x86, 0x89, 0x8C, 0x90, 0x93, 0x96,
00064     0x99, 0x9C, 0x9F, 0xA2, 0xA5, 0xA8, 0xAB, 0xAE,
00065     0xB1, 0xB3, 0xB6, 0xB9, 0xBC, 0xBF, 0xC1, 0xC4,
00066     0xC7, 0xC9, 0xCC, 0xCE, 0xD1, 0xD3, 0xD5, 0xD8,
00067     0xDA, 0xDC, 0xDE, 0xE0, 0xE2, 0xE4, 0xE6, 0xE8,
00068     0xEA, 0xEB, 0xED, 0xEF, 0xF0, 0xF1, 0xF3, 0xF4,
00069     0xF5, 0xF6, 0xF8, 0xF9, 0xFA, 0xFA, 0xFB, 0xFC,
00070     0xFD, 0xFD, 0xFE, 0xFE, 0xFE, 0xFF, 0xFF, 0xFF,
00071     0xFF, 0xFF, 0xFF, 0xFF, 0xFE, 0xFE, 0xFE, 0xFD,
00072     0xFD, 0xFC, 0xFB, 0xFA, 0xFA, 0xF9, 0xF8, 0xF6,
00073     0xF5, 0xF4, 0xF3, 0xF1, 0xF0, 0xEF, 0xED, 0xEB,
00074     0xEA, 0xE8, 0xE6, 0xE4, 0xE2, 0xE0, 0xDE, 0xDC,
00075     0xDA, 0xD8, 0xD5, 0xD3, 0xD1, 0xCE, 0xCC, 0xC9,
00076     0xC7, 0xC4, 0xC1, 0xBF, 0xBC, 0xB9, 0xB6, 0xB3,
00077     0xB1, 0xAE, 0xAB, 0xA8, 0xA5, 0xA2, 0x9F, 0x9C,
00078     0x99, 0x96, 0x93, 0x90, 0x8C, 0x89, 0x86, 0x83,
00079     0x80, 0x7D, 0x7A, 0x77, 0x74, 0x70, 0x6D, 0x6A,
00080     0x67, 0x64, 0x61, 0x5E, 0x5B, 0x58, 0x55, 0x52,
00081     0x4F, 0x4D, 0x4A, 0x47, 0x44, 0x41, 0x3F, 0x3C,
00082     0x39, 0x37, 0x34, 0x32, 0x2F, 0x2D, 0x2B, 0x28,
00083     0x26, 0x24, 0x22, 0x20, 0x1E, 0x1C, 0x1A, 0x18,
00084     0x16, 0x15, 0x13, 0x11, 0x10, 0x0F, 0x0D, 0x0C,
00085     0x0B, 0x0A, 0x08, 0x07, 0x06, 0x06, 0x05, 0x04,
00086     0x03, 0x03, 0x02, 0x02, 0x02, 0x01, 0x01, 0x01,
00087     0x01, 0x01, 0x01, 0x01, 0x02, 0x02, 0x02, 0x03,
00088     0x03, 0x04, 0x05, 0x06, 0x06, 0x07, 0x08, 0x0A,
00089     0x0B, 0x0C, 0x0D, 0x0F, 0x10, 0x11, 0x13, 0x15,
00090     0x16, 0x18, 0x1A, 0x1C, 0x1E, 0x20, 0x22, 0x24,
00091     0x26, 0x28, 0x2B, 0x2D, 0x2F, 0x32, 0x34, 0x37,
00092     0x39, 0x3C, 0x3F, 0x41, 0x44, 0x47, 0x4A, 0x4D,
00093     0x4F, 0x52, 0x55, 0x58, 0x5B, 0x5E, 0x61, 0x64,
00094     0x67, 0x6A, 0x6D, 0x70, 0x74, 0x77, 0x7A, 0x7D
00095 };
00096
00097 void led_lightshow(int speed)
00098 {
00099     static int j = 0;
00100     static uint8_t cylon = 0xff;
00101     static int alive_led = 0;
00102     static bool LED_UP = true;
00103
00104     if (j++ >= speed) { // delay a bit ok
00105         if (0) { // screen status feedback
00106             bmc.dataout.dio_buf = ~cylon; // roll leds cylon style
00107         } else {
```

```
00108             bmc.dataout.dio_buf = cylon; // roll leds cylon style (inverted)
00109         }
00110 
00111         if (LED_UP && (alive_led != 0)) {
00112             alive_led = alive_led * 2;
00113             cylon = cylon « 1;
00114         } else {
00115             if (alive_led != 0) alive_led = alive_led / 2;
00116             cylon = cylon » 1;
00117         }
00118         if (alive_led < 2) {
00119             alive_led = 2;
00120             LED_UP = true;
00121         } else {
00122             if (alive_led > 128) {
00123                 alive_led = 128;
00124                 LED_UP = false;
00125             }
00126         }
00127         j = 0;
00128     }
00129 }
00130 
00131 int main(int argc, char *argv[])
00132 {
00133     int do_ao_only = false;
00134     uint8_t i = 0, j = 75;
00135 
00136     /*
00137      * start the MQTT processing
00138      */
00139     bmc_mqtt_init();
00140 
00141     if (do_ao_only) {
00142         if (init_dac(0.0, 25.0, false) < 0) {
00143             fprintf(fout, "Missing Analog AO subdevice\n");
00144             return -1;
00145         }
00146 
00147 
00148         while (true) {
00149             set_dac_raw(0, sine_wave[255 - i++] « 4);
00150             set_dac_raw(1, sine_wave[255 - j++] « 4);
00151         }
00152     } else {
00153 #ifndef DIGITAL_ONLY
00154         if (init_daq(0.0, 25.0, false) < 0) {
00155             fprintf(fout, "Missing Analog subdevice(s)\n");
00156             return -1;
00157         }
00158 #endif
00159         if (init_dio() < 0) {
00160             fprintf(fout, "Missing Digital subdevice(s)\n");
00161             return -1;
00162         }
00163 
00164 
00165         E.dac[0] = 1.23f;
00166         E.dac[1] = 3.21f;
00167 
00168         E.do_16b = 0x01;
00169         E.di_16b = 0x10;
00170 
00171         fflush(fout);
00172         while (true) {
00173             usleep(MAIN_DELAY); // sample rate ~1 msec
00174             get_data_sample();
00175             if (!bmc.datain.D0) {
00176                 led_lightshow(0);
00177             }
00178             if (ha_flag_vars_ss.runner) { // timer or trigger from mqtt
00179                 comedi_push_mqtt(); // send json formatted data to the mqtt server
00180                 ha_flag_vars_ss.runner = false;
00181             }
00182         }
00183 
00184     }
00185     return 0;
00186 }
00187 
00188 
```

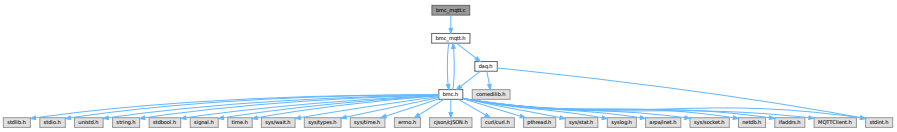
4.4 bmc.h

```
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 #include "bmc_mqtt.h"
00038
00039 #define LOG_VERSION     "V0.07"
00040 #define MQTT_VERSION    "V3.11"
00041 #define TNAME  "maint9"
00042 #define LADDRESS        "tcp://127.0.0.1:1883"
00043 #ifdef __amd64
00044 #define ADDRESS         "tcp://10.1.1.172:1883"
00045 #else
00046 #define ADDRESS         "tcp://10.1.1.172:1883"
00047 #endif
00048 #define CLIENTID1       "Energy_Mqtt_BMC1"
00049 #define CLIENTID2       "Energy_Mqtt_BMC2"
00050 #define CLIENTID3       "Energy_Mqtt_BMC3"
00051 #define TOPIC_P
00052 #define TOPIC_SPAM      "comedi/bmc/data/spam"
00053 #define TOPIC_PACA      "home-assistant/comedi/bmc"
00054         //#define TOPIC_PACB      "mateq84/data/#"
00055 #define TOPIC_AI        "comedi/bmc/data/ai"
00056 #define TOPIC_AO        "comedi/bmc/data/ao"
00057 #define TOPIC_DI        "comedi/bmc/data/di"
00058 #define TOPIC_DO        "comedi/bmc/data/do"
00059 #define QOS             1
00060
00061 #define TIMEOUT         10000L
00062 #define SPACING_USEC    500 * 1000
00063 #define USEC_SEC        1000000L
00064
00065 #define CMD_SEC         0
00066 #define CMD_USEC        10000
00067 #define TIME_SYNC_SEC   30
00068
00069 #define SBUF_SIZ        16  // short buffer string size
00070 #define RBUF_SIZ        82
00071 #define SYSLOG_SIZ      512
00072
00073 #define LOG_TO_FILE         "/var/log/bmc/bmc_comedi.log"
00074 #define LOG_TO_FILE_ALT     "/tmp/bmc_comedi.log"
00075
00076 #define MQTT_RECONN     3
00077 #define KAI             60
00078
00079 #define ANA_BUFFERS     0x40
00080
00081         /*
00082          * system testing defines
00083          * all should be undefined for normal operation
00084          */
```

```
00085         //#define DAC_TESTING
00086         //digital only
00087         //#define DIGITAL_ONLY
00088
00089         extern FILE* fout; // logging stream
00090         extern struct energy_type E;
00091
00092         struct energy_type {
00093                 volatile bool once_gti, once_ac, iammeter, fm80, dumpload, homeassistant,
      once_gti_zero, comedi;
00094                 volatile double gti_low_adj, ac_low_adj, dl_excess_adj;
00095                 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;
00096                 volatile uint32_t speed_go, im_delay, im_display, gti_delay, sequence, mqtt_count;
00097                 volatile int32_t rc, sane;
00098                 volatile uint32_t thirty_sec_clock, log_spam, log_time_reset;
00099                 pthread_mutex_t ha_lock;
00100                 volatile int16_t di_16b, do_16b;
00101                 double adc[ANA_BUFFERS], dac[ANA_BUFFERS];
00102                 MQTTClient client_p, client_sd, client_ha;
00103         };
00104
00105         void led_lightshow(int);
00106
00107 #ifdef __cplusplus
00108 }
00109 #endif
00110
00111 #endif /* BMC_H */
00112
```

4.5 bmc_mqtt.c File Reference

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



Macros

- #define COEF 12.0f

Functions

- double ac0_filter (const double)
- double ac1_filter (const double)
- 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
- struct ha_daq_hosts_type ha_daq_host

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 515 of file bmc_mqtt.c.

4.5.3.2 ac1_filter()

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

Definition at line 524 of file bmc_mqtt.c.

4.5.3.3 bmc_mqtt_init()

```
void bmc_mqtt_init (
            void )
```

Definition at line 252 of file bmc_mqtt.c.

4.5.3.4 comedi_push_mqtt()

```
void comedi_push_mqtt (
            void )
```

Definition at line 510 of file bmc_mqtt.c.

4.5.3.5 connlost()

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

trouble in River-city

Definition at line 200 of file bmc_mqtt.c.

4.5.3.6 delivered()

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

Definition at line 238 of file bmc_mqtt.c.

4.5.3.7 log_time()

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

Definition at line 164 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 390 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 313 of file bmc_mqtt.c.

4.5.3.10 showIP()

```
void showIP (
            void )
```

Definition at line 64 of file bmc_mqtt.c.

4.5.3.11 skeleton_daemon()

```
void skeleton_daemon (
            void )
```

Definition at line 105 of file bmc_mqtt.c.

4.5.3.12 timer_callback()

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

Definition at line 188 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_daq_host

```
struct ha_daq_hosts_type ha_daq_host
```

Initial value:

```
= {
    .hosts[0] = "10.1.1.30",
    .hosts[1] = "10.1.1.39",
    .hosts[2] = "10.1.2.30",
    .hosts[3] = "10.1.2.39",
    .topics[0] = "comedi/bmc/data/bmc/1",
    .topics[1] = "comedi/bmc/data/bmc/2",
    .topics[2] = "comedi/bmc/data/bmc/3",
    .topics[3] = "comedi/bmc/data/bmc/4",
    .hname[0] = "RDAQ1",
    .hname[1] = "RDAQ2",
    .hname[2] = "RDAQ3",
    .hname[3] = "RDAQ4",
    .clients[0] = "Energy_Mqtt_BMC1",
    .clients[1] = "Energy_Mqtt_BMC2",
    .clients[2] = "Energy_Mqtt_BMC3",
    .clients[3] = "Energy_Mqtt_BMC4",
    .scaler[0] = HV_SCALE0,
    .scaler[1] = HV_SCALE1,
    .scaler[2] = HV_SCALE2,
    .scaler[3] = HV_SCALE3,
    .hindex = 0,
}
```

Definition at line 33 of file bmc_mqtt.c.

4.5.4.5 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.

4.5.4.6 hname

```
char hname[256]
```

Definition at line 21 of file bmc_mqtt.c.

4.5.4.7 hname_len

```
size_t hname_len = 12
```

Definition at line 22 of file bmc_mqtt.c.

4.5.4.8 hname_ptr

```
char * hname_ptr = hname
```

Definition at line 21 of file bmc_mqtt.c.

4.5.4.9 new_timer

```
struct itimerval new_timer
```

Initial value:

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

Definition at line 8 of file bmc_mqtt.c.

4.5.4.10 old_timer

```
struct itimerval old_timer
```

Definition at line 14 of file bmc_mqtt.c.

4.5.4.11 pubmsg

```
MQTTClient_message pubmsg = MQTTClient_message_initializer
```

Definition at line 19 of file bmc_mqtt.c.

4.5.4.12 rawtime

```
time_t rawtime
```

Definition at line 15 of file bmc_mqtt.c.

4.5.4.13 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 = CMD_USEC,
00011     .it_interval.tv_sec = CMD_SEC,
00012     .it_interval.tv_usec = CMD_USEC,
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 struct ha_daq_hosts_type ha_daq_host = {
00034     .hosts[0] = "10.1.1.30",
00035     .hosts[1] = "10.1.1.39",
00036     .hosts[2] = "10.1.2.30",
00037     .hosts[3] = "10.1.2.39",
00038     .topics[0] = "comedi/bmc/data/bmc/1",
00039     .topics[1] = "comedi/bmc/data/bmc/2",
00040     .topics[2] = "comedi/bmc/data/bmc/3",
00041     .topics[3] = "comedi/bmc/data/bmc/4",
00042     .hname[0] = "RDAQ1",
00043     .hname[1] = "RDAQ2",
00044     .hname[2] = "RDAQ3",
00045     .hname[3] = "RDAQ4",
00046     .clients[0] = "Energy_Mqtt_BMC1",
00047     .clients[1] = "Energy_Mqtt_BMC2",
00048     .clients[2] = "Energy_Mqtt_BMC3",
00049     .clients[3] = "Energy_Mqtt_BMC4",
00050     .scaler[0] = HV_SCALE0,
00051     .scaler[1] = HV_SCALE1,
00052     .scaler[2] = HV_SCALE2,
00053     .scaler[3] = HV_SCALE3,
00054     .hindex = 0,
00055 };
00056
00057 double ac0_filter(const double);
00058 double ac1_filter(const double);
00059
00064 void showIP(void)
00065 {
00066     struct ifaddrs *ifaddr, *ifa;
00067     int s;
00068     char host[NI_MAXHOST];
00069
00070     if (getifaddrs(&ifaddr) == -1) {
00071         perror("getifaddrs");
00072         exit(EXIT_FAILURE);
00073     }
00074
00075
00076     for (ifa = ifaddr; ifa != NULL; ifa = ifa->ifa_next) {
00077         if (ifa->ifa_addr == NULL)
00078             continue;
00079
00080         s = getnameinfo(ifa->ifa_addr, sizeof(struct sockaddr_in), host, NI_MAXHOST, NULL, 0,
      NI_NUMERICHOST);
00081
00082         if (ifa->ifa_addr->sa_family == AF_INET) {
00083             if (s != 0) {
00084                 exit(EXIT_FAILURE);
00085             }
```

```
00086             printf("\tInterface : <%s>\n", ifa->ifa_name);
00087             printf("\t  Address : <%s>\n", host);
00088 
00089             if (strcmp(host, &ha_daq_host.hosts[0][0]) == 0) {
00090                 ha_daq_host.hindex = 0;
00091             }
00092             if (strcmp(host, &ha_daq_host.hosts[1][0]) == 0) {
00093                 ha_daq_host.hindex = 1;
00094             }
00095         }
00096     }
00097 
00098     freeifaddrs(ifaddr);
00099 }
00100 
00101 /*
00102  * setup ha_energy program to run as a background deamon
00103  * disconnect and exit foreground startup process
00104  */
00105 void skeleton_daemon(void)
00106 {
00107     pid_t pid;
00108 
00109     /* Fork off the parent process */
00110     pid = fork();
00111 
00112     /* An error occurred */
00113     if (pid < 0) {
00114         printf("\r\n%s DAEMON failure  LOG Version %s : MQTT Version %s\r\n", log_time(false),
      LOG_VERSION, MQTT_VERSION);
00115         exit(EXIT_FAILURE);
00116     }
00117 
00118     /* Success: Let the parent terminate */
00119     if (pid > 0) {
00120         exit(EXIT_SUCCESS);
00121     }
00122 
00123     /* On success: The child process becomes session leader */
00124     if (setsid() < 0) {
00125         exit(EXIT_FAILURE);
00126     }
00127 
00128     /* Catch, ignore and handle signals */
00129     /*TODO: Implement a working signal handler */
00130     //    signal(SIGCHLD, SIG_IGN);
00131     //    signal(SIGHUP, SIG_IGN);
00132 
00133     /* Fork off for the second time*/
00134     pid = fork();
00135 
00136     /* An error occurred */
00137     if (pid < 0) {
00138         exit(EXIT_FAILURE);
00139     }
00140 
00141     /* Success: Let the parent terminate */
00142     if (pid > 0) {
00143         exit(EXIT_SUCCESS);
00144     }
00145 
00146     /* Set new file permissions */
00147     umask(0);
00148 
00149     /* Change the working directory to the root directory */
00150     /* or another appropriated directory */
00151     chdir("/");
00152 
00153     /* Close all open file descriptors */
00154     int x;
00155     for (x = sysconf(_SC_OPEN_MAX); x >= 0; x--) {
00156         close(x);
00157     }
00158 
00159 }
00160 
00161 /*
00162  * sent the current UTC to the Dump Load controller
00163  */
00164 char * log_time(bool log)
00165 {
00166     static char time_log[RBUF_SIZ] = {0};
00167     time_t rawtime_log;
00168     int32_t len = 0;
00169 
00170     tzset();
00171     timezone = 0;
```

```
00172     daylight = 0;
00173     time(&rawtime_log);
00174     sprintf(time_log, "%s", ctime(&rawtime_log));
00175     len = strlen(time_log);
00176     time_log[len - 1] = 0; // munge out the return character
00177     if (log) {
00178         fprintf(fout, "%s ", time_log);
00179         fflush(fout);
00180     }
00181     return time_log;
00182 }
00183
00184 /*
00185  * data update timer flag
00186  * and CMD_SEC seconds software time clock
00187  */
00188 void timer_callback(int32_t signum)
00189 {
00190     signal(signum, timer_callback);
00191     ha_flag_vars_ss.runner = true;
00192     E.thirty_sec_clock++;
00193     E.log_time_reset++;
00194
00195 }
00196
00197 /*
00198  * MQTT Broker connection errors can be fatal
00199  */
00200 void connlost(void *context, char *cause)
00201 {
00202     struct ha_flag_type *ha_flag = context;
00203     int32_t id_num = ha_flag->ha_id;
00204     static uint32_t times = 0;
00205     char * where = "Missing Topic";
00206     char * what = "Reconnection Error";
00207
00208     // bug-out if no context variables passed to callback
00209     if (context == NULL) {
00210         id_num = -1;
00211         goto bugout;
00212     }
00213
00214     if (times++ > MQTT_RECONN) {
00215         goto bugout;
00216     } else {
00217         if (times > 1) {
00218             fprintf(fout, "%s Connection lost, retrying %d \n", log_time(false), times);
00219             fprintf(fout, "%s     cause: %s, h_id %d, c_id %d, %s \n", log_time(false), cause, id_num,
      0, what);
00220             fprintf(fout, "%s MQTT DAEMON reconnect failure,  LOG Version %s : MQTT Version %s\n",
      log_time(false), LOG_VERSION, MQTT_VERSION);
00221         }
00222         fflush(fout);
00223         times = 0;
00224         return;
00225     }
00226
00227 bugout:
00228     fprintf(fout, "%s Connection lost, exit ha_energy program\n", log_time(false));
00229     fprintf(fout, "%s     cause: %s, h_id %d, c_id %d, %s \n", log_time(false), cause, id_num, 0,
      where);
00230     fprintf(fout, "%s MQTT DAEMON context is NULL failure,  LOG Version %s : MQTT Version %s\n",
      log_time(false), LOG_VERSION, MQTT_VERSION);
00231     fflush(fout);
00232     exit(EXIT_FAILURE);
00233 }
00234
00235 /*
00236  * set the broker has message token
00237  */
00238 void delivered(void *context, MQTTClient_deliveryToken dt)
00239 {
00240     struct ha_flag_type *ha_flag = context;
00241
00242     // bug-out if no context variables passed to callback
00243     if (context == NULL) {
00244         return;
00245     }
00246     ha_flag->deliveredtoken = dt;
00247 }
00248
00252 void bmc_mqtt_init(void)
00253 {
00254     E.mqtt_count = 0;
00255     gethostname(hname, hname_len);
00256     hname[12] = 0;
00257     printf("\r\n  LOG Version %s : MQTT Version %s : Host Name %s\r\n", LOG_VERSION, MQTT_VERSION,
```

```
    hname);
00258     showIP();
00259     skeleton_daemon();
00260 #ifdef LOG_TO_FILE
00261     fout = fopen(LOG_TO_FILE, "a");
00262     if (fout == NULL) {
00263         fout = fopen(LOG_TO_FILE_ALT, "a");
00264         if (fout == NULL) {
00265             fout = stdout;
00266             printf("\r\n%s Unable to open LOG file %s \r\n", log_time(false), LOG_TO_FILE_ALT);
00267         }
00268     }
00269 #else
00270     fout = stdout;
00271 #endif
00272
00273     /*
00274      * set the timer for MQTT publishing sample speed
00275      * CMD_SEC sets the time in seconds
00276      */
00277     setitimer(ITIMER_REAL, &new_timer, &old_timer);
00278     signal(SIGALRM, timer_callback);
00279
00280     if (strncmp(hname, TNAME, 6) == 0) {
00281         MQTTClient_create(&E.client_p, LADDRESS, (const char *)
    &ha_daq_host.topics[ha_daq_host.hindex],
00282             MQTTCLIENT_PERSISTENCE_NONE, NULL);
00283         conn_opts_p.keepAliveInterval = KAI;
00284         conn_opts_p.cleansession = 1;
00285         hname_ptr = LADDRESS;
00286     } else {
00287         MQTTClient_create(&E.client_p, ADDRESS, (const char *)
    &ha_daq_host.clients[ha_daq_host.hindex],
00288             MQTTCLIENT_PERSISTENCE_NONE, NULL);
00289         conn_opts_p.keepAliveInterval = KAI;
00290         conn_opts_p.cleansession = 1;
00291         hname_ptr = ADDRESS;
00292     }
00293
00294     fprintf(fout, "\r\n%s Connect MQTT server %s, %s\n", log_time(false), hname_ptr, (const char *)
    &ha_daq_host.clients[ha_daq_host.hindex]);
00295     fflush(fout);
00296     MQTTClient_setCallbacks(E.client_p, &ha_flag_vars_ss, connlost, msgarrvd, delivered);
00297     if ((E.rc = MQTTClient_connect(E.client_p, &conn_opts_p)) != MQTTCLIENT_SUCCESS) {
00298         fprintf(fout, "%s Failed to connect MQTT server, return code %d %s, %s\n", log_time(false),
    E.rc, hname_ptr, (const char *) &ha_daq_host.clients[ha_daq_host.hindex]);
00299         fflush(fout);
00300         pthread_mutex_destroy(&E.ha_lock);
00301         exit(EXIT_FAILURE);
00302     }
00303
00304     MQTTClient_subscribe(E.client_p, ha_daq_host.topics[ha_daq_host.hindex], QOS); // remote DAQ data
    for the HA_Energy system
00305
00306     pubmsg.payload = "online";
00307     pubmsg.payloadlen = strlen("online");
00308     pubmsg.qos = QOS;
00309     pubmsg.retained = 0;
00310     ha_flag_vars_ss.deliveredtoken = 0;
00311 }
00312
00313 int32_t msgarrvd(void *context, char *topicName, int topicLen, MQTTClient_message *message)
00314 {
00315     int32_t i, ret = 1;
00316     const char* payloadptr;
00317     char buffer[MBMQTT];
00318     struct ha_flag_type *ha_flag = context;
00319
00320     E.mqtt_count++;
00321     // bug-out if no context variables passed to callback
00322     if (context == NULL) {
00323         ret = -1;
00324         goto null_exit;
00325     }
00326
00327 #ifdef DEBUG_REC
00328     fprintf(fout, "Message arrived\n");
00329 #endif
00330     /*
00331      * move the received message into a processing holding buffer
00332      */
00333     payloadptr = message->payload;
00334     for (i = 0; i < message->payloadlen; i++) {
00335         buffer[i] = *payloadptr++;
00336     }
00337     buffer[i] = 0; // make a null terminated C string
00338
```

```
00339     // parse the JSON data in the holding buffer
00340     cJSON *json = cJSON_ParseWithLength(buffer, message->payloadlen);
00341     if (json == NULL) {
00342         const char *error_ptr = cJSON_GetErrorPtr();
00343         if (error_ptr != NULL) {
00344             fprintf(fout, "%s Error: %s NULL cJSON pointer\n", log_time(false), error_ptr);
00345         }
00346         ret = -1;
00347         ha_flag->rec_ok = false;
00348         E.comedi = false;
00349         goto error_exit;
00350     }
00351
00352     /*
00353      * MQTT messages for COMEDI
00354      */
00355 #ifdef DEBUG_REC
00356     fprintf(fout, "COMEDI MQTT data\r\n");
00357 #endif
00358     cJSON *data_result = json;
00359
00360     data_result = cJSON_GetObjectItemCaseSensitive(json, "Comedi_Request");
00361
00362     if (cJSON_IsString(data_result) && (data_result->valuestring != NULL)) {
00363         fprintf(fout, "%s Comedi Trigger from MQTT server, Topic %s %s\n", log_time(false), topicName,
    data_result->valuestring);
00364         fflush(fout);
00365         ret = true;
00366     }
00367     E.comedi = true;
00368
00369     // done with processing MQTT async message, set state flags
00370     ha_flag->receivedtoken = true;
00371     ha_flag->rec_ok = true;
00372     ha_flag_vars_ss.runner = true; // send data in response to received message of any type
00373     /*
00374      * exit and delete/free resources. In steps depending of possible error conditions
00375      */
00376 error_exit:
00377     // delete the JSON object
00378     cJSON_Delete(json);
00379 null_exit:
00380     // free the MQTT objects
00381     MQTTClient_freeMessage(&message);
00382     MQTTClient_free(topicName);
00383     fflush(fout);
00384     return ret;
00385 }
00386
00387 /*
00388  * send Comedi variables MQTT host
00389  */
00390 void mqtt_bmc_data(MQTTClient client_p, const char * topic_p)
00391 {
00392     cJSON *json;
00393     time_t rawtime;
00394     static uint32_t spam = 0;
00395     double over_sample;
00396     static uint32_t pacer = 0;
00397
00398     MQTTClient_message pubmsg = MQTTClient_message_initializer;
00399     MQTTClient_deliveryToken token;
00400     ha_flag_vars_ss.deliveredtoken = 0;
00401
00402     fprintf(fout, "%s Sending Comedi data to MQTT server, Topic %s\n", log_time(false), topic_p);
00403     fflush(fout);
00404
00405 #ifndef DIGITAL_ONLY
00406     over_sample = 0.0f; // over-sample avg
00407     for (int i = 0; i < OVER_SAMP; i++) {
00408         if (bmc.BOARD == bmcboard) {
00409             over_sample += ac0_filter(get_adc_volts(channel_ANA1));
00410         } else {
00411             over_sample += ac0_filter(get_adc_volts(0));
00412         }
00413     }
00414     E.adc[0] = over_sample / (double) OVER_SAMP;
00415
00416     over_sample = 0.0f; // over-sample avg
00417     for (int i = 0; i < OVER_SAMP; i++) {
00418         if (bmc.BOARD == bmcboard) {
00419             over_sample += ac1_filter(get_adc_volts(channel_ANA2));
00420         } else {
00421             over_sample += ac1_filter(get_adc_volts(1));
00422         }
00423     }
00424     E.adc[1] = over_sample / (double) OVER_SAMP;
```

```
00425     if (bmc.BOARD == bmcboard) {
00426         E.adc[channel_ANA4] = get_adc_volts(channel_ANA4);
00427         E.adc[channel_ANA5] = get_adc_volts(channel_ANA5);
00428         E.adc[channel_ANC6] = get_adc_volts(channel_ANC6);
00429         E.adc[channel_ANC7] = get_adc_volts(channel_ANC7);
00430     }
00431
00432 #ifdef DAC_TESTING
00433     E.dac[0] = E.adc[0];
00434     E.dac[1] = E.adc[1];
00435 #endif
00436
00437 #ifndef DAC_TESTING
00438     set_dac_raw(0, 0);
00439     set_dac_raw(1, 0);
00440 #else
00441     set_dac_volts(0, E.dac[0]);
00442     set_dac_volts(1, E.dac[1]);
00443 #endif
00444 #endif
00445
00446     E.do_16b = bmc.dataout.dio_buf;
00447     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);
00448
00449     if (pacer++ > 100) {
00450         pacer = 0;
00451         E.mqtt_count++;
00452         E.sequence++;
00453         json = cJSON_CreateObject();
00454         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "name", 64);
00455         cJSON_AddStringToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], (const
      char *) &ha_daq_host.clients[ha_daq_host.hindex]);
00456         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "board", 64);
00457         cJSON_AddStringToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], (const
      char *) bmc.BNAME);
00458         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "sequence", 64);
00459         cJSON_AddNumberToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex],
      E.sequence);
00460         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "mqtt_do_16b", 64);
00461         cJSON_AddNumberToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], (double)
      E.do_16b);
00462         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "mqtt_di_16b", 64);
00463         cJSON_AddNumberToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], (double)
      E.di_16b);
00464         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "bmc_adc0", 64);
00465         cJSON_AddNumberToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex],
      E.adc[0]);
00466         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "bmc_adc1", 64);
00467         cJSON_AddNumberToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex],
      E.adc[1]);
00468         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "build_date", 64);
00469         cJSON_AddStringToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], FW_Date);
00470         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "build_time", 64);
00471         cJSON_AddStringToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], FW_Time);
00472         time(&rawtime);
00473         strncpy(&ha_daq_host.hname[ha_daq_host.hindex][5], "sequence_time", 64);
00474         cJSON_AddNumberToObject(json, (const char *) &ha_daq_host.hname[ha_daq_host.hindex], (double)
      rawtime);
00475         // convert the cJSON object to a JSON string
00476         char *json_str = cJSON_Print(json);
00477
00478         pubmsg.payload = json_str;
00479         pubmsg.payloadlen = strlen(json_str);
00480         pubmsg.qos = QOS;
00481         pubmsg.retained = 0;
00482
00483         MQTTClient_publishMessage(client_p, topic_p, &pubmsg, &token);
00484         // a busy, wait loop for the async delivery thread to complete
00485         {
00486             uint32_t waiting = 0;
00487             while (ha_flag_vars_ss.deliveredtoken != token) {
00488                 usleep(TOKEN_DELAY);
00489                 if (waiting++ > MQTT_RETRY) {
00490                     if (spam++ > 1) {
00491                         fprintf(fout, "%s SW mqtt_bmc_data, Still Waiting, timeout\r\n",
      log_time(false));
00492                         fflush(fout);
00493                         spam = 0;
00494                     }
00495                     break;
00496                 } else {
00497                     spam = 0;
00498                 }
00499             };
00500         }
00501
```

```
00502         cJSON_free(json_str);
00503         cJSON_Delete(json);
00504     }
00505 }
00506
00507 /*
00508  * main program function to send Comedi data to the MQTT server
00509  */
00510 void comedi_push_mqtt(void)
00511 {
00512     mqtt_bmc_data(E.client_p, ha_daq_host.topics[ha_daq_host.hindex]);
00513 }
00514
00515 double ac0_filter(const double raw)
00516 {
00517     static double accum = 0.0f;
00518     ;
00519     static double coef = COEF;
00520     accum = accum - accum / coef + raw;
00521     return accum / coef;
00522 }
00523
00524 double ac1_filter(const double raw)
00525 {
00526     static double accum = 0.0f;
00527     static double coef = COEF;
00528     accum = accum - accum / coef + raw;
00529     return accum / coef;
00530 }
```

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
00012 #define MQTT_RETRY 10
00013
00014 #define HA_SW_DELAY     400000  // usecs
00015 #define TOKEN_DELAY     600
00016 #define GTI_TOKEN_DELAY 300
00017
00018 #define MAIN_DELAY      1000 // 1msec comedi sample rate max
00019
00020 #define QOS             1
00021
00022 #define RDEV_SIZE        10
00023
00024 #define SLEEP_CODE      0
00025 #define FLOAT_CODE      1
00026
00027 #define MBMQTT  1024
00028
00029         enum mqtt_id {
00030                 P8055_ID,
00031                 FM80_ID,
00032                 DUMPLOAD_ID,
00033                 HA_ID,
00034                 COMEDI_ID,
00035                 LAST_MQTT_ID,
00036         };
00037
00038         struct ha_flag_type {
00039                 volatile MQTTClient_deliveryToken deliveredtoken, receivedtoken;
00040                 volatile bool runner, rec_ok;
00041                 int32_t ha_id;
00042                 volatile int32_t var_update, energy_mode;
00043         };
00044
00045         struct ha_daq_hosts_type {
00046                 const char hosts[4][NI_MAXHOST];
00047                 const char clients[4][NI_MAXHOST];
00048                 const char topics[4][NI_MAXHOST];
00049                 char hname[4][NI_MAXHOST];
00050                 double scaler[4];
00051                 uint8_t hindex;
```

```
00052         };
00053
00054         extern struct ha_flag_type ha_flag_vars_ss, ha_daq_hosts_type;
00055         extern struct ha_daq_hosts_type ha_daq_host;
00056
00057         void mqtt_bmc_data(MQTTClient, const char *);
00058         void delivered(void *, MQTTClient_deliveryToken);
00059         int32_t msgarrvd(void *, char *, int, MQTTClient_message *);
00060         void connlost(void *, char *);
00061         void showIP(void);
00062         void skeleton_daemon(void);
00063         void bmc_mqtt_init(void);
00064         char * log_time(bool);
00065
00066         void timer_callback(int32_t);
00067         void comedi_push_mqtt(void);
00068
00069 #ifdef __cplusplus
00070 }
00071 #endif
00072
00073 #endif /* BMC_MQTT_H */
00074
```

4.8 bmcdio.c

```
00001 #include "bmcdio.h"
00002
00003 static wchar_t wstr[MAX_STR]; // buffer for id settings strings from MPC2210
00004
00005 static mcp2210_spi_type SPI_buffer; //  MCP2210 I/O structure
00006 mcp2210_spi_type *S = &SPI_buffer; // working I/O structure pointer
00007 static const char *build_date = __DATE__, *build_time = __TIME__;
00008
00009 /*
00010  * zero rx/tx buffers
00011  */
00012 void cbufs(void)
00013 {
00014     memset(S->buf, 0, sizeof(S->buf)); // initialize bufs to zeros
00015     memset(S->rbuf, 0, sizeof(S->rbuf));
00016 }
00017
00018 /*
00019  * send and receive USB data
00020  * using the asynchronous hid device mode for this driver
00021  */
00022 int32_t SendUSBCmd(hid_device *handle, uint8_t *cmdBuf, uint8_t *responseBuf)
00023 {
00024     int32_t r;
00025
00026     r = hid_write(handle, cmdBuf, COMMAND_BUFFER_LENGTH);
00027     if (r < 0) {
00028         return ERROR_UNABLE_TO_WRITE_TO_DEVICE;
00029     }
00030
00031     //when the hid device is configured as synchronous, the first
00032     //hid_read returns the desired results. and the while() loop
00033     //is skipped.
00034     //
00035     //when the hid device is configured as asynchronous, the first
00036     //hid_read may or may not succeed, depending on the latency
00037     //of the attached device. When no data is returned, r = 0 and
00038     //the while loop keeps polling the returned data until it is
00039     //received.
00040     r = hid_read(handle, responseBuf, RESPONSE_BUFFER_LENGTH);
00041     if (r < 0) {
00042         return ERROR_UNABLE_TO_READ_FROM_DEVICE;
00043     }
00044
00045     while (r == 0) {
00046         r = hid_read(handle, responseBuf, RESPONSE_BUFFER_LENGTH);
00047         if (r < 0) {
00048             return ERROR_UNABLE_TO_READ_FROM_DEVICE;
00049         }
00050         sleep_us(SPI_STATUS_DELAY_US);
00051     }
00052
00053     return responseBuf[1];
00054 }
00055
00056 int32_t get_usb_res(void)
00057 {
```

```
00058     return S->res;
00059 }
00060
00061 int32_t nanosleep(const struct timespec *, struct timespec *);
00062
00063 void sleep_us(const uint32_t microseconds)
00064 {
00065     struct timespec ts;
00066     if (!microseconds) {
00067         return;
00068     }
00069     ts.tv_sec = microseconds / 1000000; // whole seconds
00070     ts.tv_nsec = (microseconds % 1000000) * 1000; // remainder, in nanoseconds
00071     nanosleep(&ts, NULL);
00072 }
00073
00074 /*
00075  * when connected to the TIC12400 interrupt pin it shows a switch has changed state
00076  */
00077 bool get_MCP2210_ext_interrupt(void)
00078 {
00079 #ifdef EXT_INT_DPRINT
00080     static uint32_t counts = 0;
00081 #endif
00082
00083     cbufs();
00084     S->buf[0] = 0x12; // Get (VM) the Current Number of Events From the Interrupt Pin, GPIO 6 FUNC2
00085     S->buf[1] = 0x00; // reads, then resets the event counter
00086     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00087     if (S->rbuf[4] || S->rbuf[5]) {
00088 #ifdef EXT_INT_DPRINT
00089         printf("\r\nrbuf4 %x: rbuf5 %x: counts %i\n", S->rbuf[4], S->rbuf[5], ++counts);
00090 #endif
00091         return true;
00092     }
00093     return false;
00094 }
00095
00096 int32_t cancel_spi_transfer(void)
00097 {
00098     cbufs();
00099     S->buf[0] = 0x11; // 0x11 cancel SPI transfer
00100     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00101     return S->res;
00102 }
00103
00104 bool SPI_WriteRead(hid_device *handle, uint8_t *buf, uint8_t *rbuf)
00105 {
00106     S->res = SendUSBCmd(handle, buf, rbuf);
00107     while (rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || rbuf[3] == SPI_STATUS_SUCCESSFUL) {
00108         S->res = SendUSBCmd(handle, buf, rbuf);
00109     }
00110     return true;
00111 }
00112
00113 bool SPI_MCP2210_WriteRead(uint8_t* pTransmitData, const size_t txSize, uint8_t* pReceiveData, const
    size_t rxSize)
00114 {
00115 #ifdef DPRINT
00116     static uint32_t tx_count = 0;
00117 #endif
00118
00119     cbufs();
00120     S->buf[0] = 0x42; // transfer SPI data command
00121     S->buf[1] = rxSize; // no. of SPI bytes to transfer
00122     S->buf[4] = pTransmitData[3];
00123     S->buf[5] = pTransmitData[2];
00124     S->buf[6] = pTransmitData[1];
00125     S->buf[7] = pTransmitData[0];
00126     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00127 #ifdef DPRINT
00128     printf("TX SPI res %x - tx count %i\n", S->res, ++tx_count);
00129 #endif
00130
00131 #ifdef DPRINT
00132     uint32_t rcount = 0;
00133 #endif
00134     while (S->rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || S->rbuf[3] == SPI_STATUS_SUCCESSFUL)
    {
00135 #ifdef DPRINT
00136         printf("SPI RX wait %i: code %x\n", ++rcount, S->rbuf[3]);
00137 #endif
00138         S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00139     }
00140     pReceiveData[3] = S->rbuf[4];
00141     pReceiveData[2] = S->rbuf[5];
00142     pReceiveData[1] = S->rbuf[6];
```

```
00143     pReceiveData[0] = S->rbuf[7];
00144     return true;
00145 }
00146
00147 /*
00148  * open the USB device with a optional serial number
00149  */
00150 mcp2210_spi_type* hidrawapi_mcp2210_init(const wchar_t *serial_number)
00151 {
00152     printf("\r\n--- MCP2210 Driver Version %s %s %s ---\r\n", MCP2210_DRIVER, build_date, build_time);
00153     cbufs(); // clear command and response buffers
00154     //------------------ Open MCP2210 and display info ---------------
00155     printf("Open Device: 04d8:00de\n");
00156     // Open the device using the VID(vendor ID, PID(product ID),
00157     // and optionally the Serial number.
00158     S->handle = hid_open(0x4d8, 0xde, serial_number); // open the MCP2210 device
00159     if (!S->handle) {
00160         printf("unable to open the MCP2210\n");
00161         return NULL;
00162     }
00163     // Read the Manufacturer String
00164     wstr[0] = 0x0000;
00165     S->res = hid_get_manufacturer_string(S->handle, wstr, MAX_STR);
00166     if (S->res < 0) {
00167         printf("Unable to read manufacturer string\n");
00168     }
00169     printf("Manufacturer String: %ls\n", wstr);
00170
00171     // Read the Product String
00172     wstr[0] = 0x0000;
00173     S->res = hid_get_product_string(S->handle, wstr, MAX_STR);
00174     if (S->res < 0) {
00175         printf("Unable to read product string\n");
00176     }
00177     printf("Product String: %ls\n", wstr);
00178
00179     // Read the Serial Number String
00180     wstr[0] = 0x0000;
00181     S->res = hid_get_serial_number_string(S->handle, wstr, MAX_STR);
00182     if (S->res < 0) {
00183         printf("Unable to read serial number string\n");
00184     }
00185     printf("Serial Number String: (%d) %ls", wstr[0], wstr);
00186     printf("\n");
00187
00188     hid_set_nonblocking(S->handle, 1); // async operation, don't block
00189
00190     //-------------- Set GPIO pin function (0x21) -------------
00191     // configure chip selects and interrupts for all devices on the SPI buss
00192     cbufs();
00193     S->buf[0] = 0x21; // command 21 - set GPIO pin's functions
00194     S->buf[4] = 0x01; // GPIO 0  set to 0x01 - SPI CS, BMX160
00195     S->buf[5] = 0x00; // GPIO 1
00196     S->buf[6] = 0x00; // GPIO 2
00197     S->buf[7] = 0x02; // GPIO 3, act led
00198     S->buf[8] = 0x02; // GPIO 4, LOWPWR led
00199     S->buf[9] = 0x01; // GPIO 5 set to 0x01 - SPI CS, tic12400
00200     S->buf[10] = 0x02; // GPIO 6 external interrupt input
00201     S->buf[11] = 0x01; // GPIO 7 set to 0x01 - SPI CS, MC33996
00202     S->buf[12] = 0x00; // GPIO 8
00203     S->buf[13] = 0xff;
00204     S->buf[14] = 0x01;
00205     S->buf[15] = 0b01000000; // set GPIO 6 to input
00206     S->buf[16] = 0b00000001; //
00207     S->buf[17] = 0b00000010; // count Falling edges
00208     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00209
00210     // ------------ Set GPIO pin direction (0x32)--------------
00211     cbufs();
00212     S->buf[0] = 0x32; // command 32 - set GPIO pin's directions
00213     // function:  0 = output, 1 = input
00214     S->buf[4] = 0b01000000; // set GPIO 0-5,7 to outputs, GPIO 6 for input
00215     S->buf[5] = 0b00000001; // set GPIO 8 to input
00216     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00217
00218     // ------------ Set GPIO pin level (0x30)--------------
00219     cbufs();
00220     S->buf[0] = 0x30; // command 30 - set GPIO pin's level to all high
00221     S->buf[4] = 0xff;
00222     S->buf[5] = 0xff;
00223     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00224
00225     cancel_spi_transfer(); // reset the SPI engine
00226     printf("MCP2210 init complete\n"); // ctrl c to exit program
00227     return S;
00228 }
00229
```

```
00230 /*
00231  * read SPI data from BMX160 register
00232  */
00233 uint8_t bmx160_get(uint8_t nbytes, uint8_t addr)
00234 {
00235     cbufs();
00236     // BMX160 config
00237     S->buf[0] = 0x42; // transfer SPI data command
00238     S->buf[1] = nbytes; // no. of SPI bytes to transfer
00239     S->buf[4] = addr | BMX160_R; //device address, read
00240     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00241     while (S->rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || S->rbuf[3] == SPI_STATUS_SUCCESSFUL)
    {
00242         S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00243     }
00244     return S->rbuf[5];
00245 }
00246
00247 /*
00248  * write SPI data to BMX160 register
00249  */
00250 uint8_t bmx160_set(uint8_t set_data, uint8_t addr)
00251 {
00252     cbufs();
00253     // BMX160 config
00254     S->buf[0] = 0x42; // transfer SPI data command
00255     S->buf[1] = 2; // no. of SPI bytes to transfer
00256     S->buf[4] = addr | BMX160_W; //device address, write
00257     S->buf[5] = set_data;
00258     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00259     while (S->rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || S->rbuf[3] == SPI_STATUS_SUCCESSFUL)
    {
00260         S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00261     }
00262     return S->rbuf[5];
00263 }
00264
00265 /*
00266  * write SPI data to 3 BMX160 registers
00267  */
00268 uint8_t bmx160_set3(uint8_t *set_data, uint8_t addr)
00269 {
00270     cbufs();
00271     // BMX160 config
00272     S->buf[0] = 0x42; // transfer SPI data command
00273     S->buf[1] = 4; // no. of SPI bytes to transfer
00274     S->buf[4] = addr | BMX160_W; //device address, write
00275     S->buf[5] = set_data[0];
00276     S->buf[6] = set_data[1];
00277     S->buf[7] = set_data[2];
00278     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00279     while (S->rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || S->rbuf[3] == SPI_STATUS_SUCCESSFUL)
    {
00280         S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00281     }
00282     return S->rbuf[5];
00283 }
00284
00285 /*
00286  * USB to SPI configuration for BMX160
00287  */
00288 void setup_bmx160_transfer(uint8_t nbytes)
00289 {
00290     cbufs();
00291     S->buf[0] = 0x40; // SPI transfer settings command
00292     S->buf[4] = 0x00; // set SPI transfer bit rate;
00293     S->buf[5] = 0x09; // 32 bits, lsb = buf[4], msb buf[7]
00294     S->buf[6] = 0x3D; // 4MHz
00295     S->buf[7] = 0x00;
00296     S->buf[8] = 0xff; // set CS idle values to 1
00297     S->buf[9] = 0x01;
00298     S->buf[10] = 0b11111110; // set CS active values to 0, set the rest to 1
00299     S->buf[11] = 0b11111111;
00300     S->buf[18] = nbytes; // set no of bytes to transfer = 3
00301     S->buf[20] = 0x03; // spi mode 3
00302     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00303 }
00304
00305 void setup_tic12400_transfer(void)
00306 {
00307     cbufs();
00308     S->buf[0] = 0x40; // SPI transfer settings command
00309     S->buf[4] = 0x00; // set SPI transfer bit rate;
00310     S->buf[5] = 0x09; // 32 bits, lsb = buf[4], msb buf[7]
00311     S->buf[6] = 0x3d; // 4MHz
00312     S->buf[7] = 0x00;
00313     S->buf[8] = 0xff; // set CS idle values to 1
```

```
00314     S->buf[9] = 0x01;
00315     S->buf[10] = 0b11011111; // set CS active values to 0, set the rest to 1
00316     S->buf[11] = 0b11111111;
00317     S->buf[18] = 0x4; // set no of bytes to transfer = 4 // 32-bit transfers
00318     S->buf[20] = 0x01; // spi mode 1
00319     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00320 }
00321
00322 void get_bmx160_transfer(void)
00323 {
00324     // ---------- Get SPI transfer settings (0x41)-------------
00325     cbufs();
00326     S->buf[0] = 0x41; // 0x41 Get SPI transfer settings
00327     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00328     printf("SPI BMX160 transfer settings\n   "); // Print out the 0x41 returned buffer.
00329     for (int i = 0; i < S->rbuf[2]; i++) {
00330         printf("%02hhx ", S->rbuf[i]);
00331     }
00332     printf("\n");
00333 }
00334
00335 /*
00336  * read raw BMX160 sensor data array and print to stdio
00337  */
00338 void show_bmx160_transfer(void)
00339 {
00340     printf("SPI BMX160 IMU data   "); // Print out the 0x41 returned buffer.
00341     for (int i = 5; i < 28; i++) {
00342         printf("%02hhx ", S->rbuf[i]);
00343     }
00344     printf("\r");
00345 }
00346
00347 /*
00348  * get raw sensor data from IMU and transfer to buffer
00349  */
00350 void move_bmx160_transfer_data(uint8_t *pBuf)
00351 {
00352     if (pBuf) {
00353         for (int i = 5; i < 28; i++) {
00354             pBuf[i - 5] = S->rbuf[i];
00355         }
00356     }
00357 }
00358
00359 /*
00360  * get raw sensor status 0x1B from IMU and transfer to buffer
00361  */
00362 void move_bmx160_transfer_status(uint8_t *pBuf)
00363 {
00364     if (pBuf) {
00365         for (int i = 28; i < 36; i++) {
00366             pBuf[i - 28] = S->rbuf[i];
00367         }
00368     }
00369 }
00370
00371 /*
00372  *
00373  */
00374 void bmx160_update(void)
00375 {
00376 }
00377
00378 void get_tic12400_transfer(void)
00379 {
00380     // ---------- Get SPI transfer settings (0x41)-------------
00381     cbufs();
00382     S->buf[0] = 0x41; // 0x41 Get SPI transfer settings
00383     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00384     printf("SPI TIC12400 transfer settings\n   "); // Print out the 0x41 returned buffer.
00385     for (int i = 0; i < S->rbuf[2]; i++) {
00386         printf("%02hhx ", S->rbuf[i]);
00387     }
00388     printf("\n");
00389 }
00390
00391 /*
00392  * config the SPI device outputs to a default condition
00393  */
00394 void mc33996_init(void)
00395 {
00396     cbufs(); // clear the RX/TX buffer
00397     // MCP33996 config
00398     S->buf[0] = 0x42; // transfer SPI data command
00399     S->buf[1] = 3; // no. of SPI bytes to transfer
00400     S->buf[4] = 0x00; // on/off control
```

```
00401     S->buf[5] = 0x0f; // set all outputs to low
00402     S->buf[6] = 0xf0; // ""
00403     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00404     while (S->rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || S->rbuf[3] == SPI_STATUS_SUCCESSFUL)
    {
00405         S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00406     }
00407 };
00408
00409 /*
00410  * update the SPI device outputs
00411  */
00412 void mc33996_set(uint8_t cmd, uint8_t data_h, uint8_t data_l)
00413 {
00414     cbufs(); // clear the RX/TX buffer
00415     // MCP33996 config
00416     S->buf[0] = 0x42; // transfer SPI data command
00417     S->buf[1] = 6; // no. of SPI bytes to transfer
00418     S->buf[4] = cmd; // device command control
00419     S->buf[5] = data_h; // set all outputs
00420     S->buf[6] = data_l; // ""
00421     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00422     while (S->rbuf[3] == SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE || S->rbuf[3] == SPI_STATUS_SUCCESSFUL)
    {
00423         S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00424     }
00425 };
00426
00427 /*
00428  * check for correct 48-bit data from mc33996 after command reset
00429  */
00430 bool mc33996_check(void)
00431 {
00432     if (S->rbuf[7] == mc33996_reset && S->rbuf[8] == mc33996_magic_h && S->rbuf[9] == mc33996_magic_l)
    {
00433         return true;
00434     } else {
00435         return false;
00436     }
00437 };
00438
00439 /*
00440  * config the USB to SPI parameters for the slave device
00441  */
00442 void setup_mc33996_transfer(uint8_t len)
00443 {
00444     cbufs();
00445     S->buf[0] = 0x40; // SPI transfer settings command
00446     S->buf[4] = 0x00; // set SPI transfer bit rate;
00447     S->buf[5] = 0x09; // 32 bits, lsb = buf[4], msb buf[7]
00448     S->buf[6] = 0x3d; // 4MHz
00449     S->buf[7] = 0x00; // ""
00450     S->buf[8] = 0xff; // set CS idle values to 1
00451     S->buf[9] = 0x01; // ""
00452     S->buf[10] = 0b01111111; // set CS active values to 0, set the rest to 1
00453     S->buf[11] = 0b11111111; // ""
00454     S->buf[18] = len; // set no of bytes to transfer = 3
00455     S->buf[20] = 0x01; // spi mode 1
00456     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00457 };
00458
00459 /*
00460  * display to stdio the USB to SPI transfer parameters
00461  */
00462 void get_mc33996_transfer(void)
00463 {
00464     cbufs();
00465     S->buf[0] = 0x41; // 0x41 Get SPI transfer settings
00466     S->res = SendUSBCmd(S->handle, S->buf, S->rbuf);
00467     printf("SPI MC33996 transfer settings\n   "); // Print out the 0x41 returned buffer.
00468     for (int i = 0; i < S->rbuf[2]; i++) {
00469         printf("%02hhx ", S->rbuf[i]);
00470     }
00471     printf("\n");
00472 };
00473
00474 /*
00475  * setup SPI command sequence
00476  */
00477 void mc33996_update(void)
00478 {
00479     S->buf[4] = mc33996_control; // set MC33996 outputs command
00480 };
```

4.9 bmcdio.h

```
00001 /*
00002  * File:   bmcdio.h
00003  * Author: root
00004  *
00005  * Created on May 12, 2025, 11:20AM
00006  */
00007
00008 #ifndef BMCDIO_H
00009 #define BMCDIO_H
00010
00011 #ifdef __cplusplus
00012 extern "C" {
00013 #endif
00014
00015 #include <stdlib.h>
00016 #include <stdbool.h>
00017 #include <stdint.h>
00018 #include <stdio.h>
00019 #include <wchar.h>
00020 #include <string.h>
00021 #include <fcntl.h>           /* Definition of AT_* constants */
00022 #include <unistd.h>
00023 #include <time.h>
00024 #include <sys/stat.h>
00025 #include <sys/types.h>
00026 #include <hidapi/hidapi.h>
00027 #include "tic12400.h"
00028 #include "mc33996.h"
00029
00030 #define MCP2210_DRIVER "V0.8"
00031
00032 #define MAX_STR                                 255
00033 #define OPERATION_SUCCESSFUL                    0
00034 #define ERROR_UNABLE_TO_OPEN_DEVICE             -1
00035 #define ERROR_UNABLE_TO_WRITE_TO_DEVICE         -2
00036 #define ERROR_UNABLE_TO_READ_FROM_DEVICE        -3
00037 #define ERROR_INVALID_DEVICE_HANDLE             -99
00038
00039 #define COMMAND_BUFFER_LENGTH                   64
00040 #define RESPONSE_BUFFER_LENGTH                  64
00041
00042 #define SPI_STATUS_FINISHED_NO_DATA_TO_SEND     0x10
00043 #define SPI_STATUS_STARTED_NO_DATA_TO_RECEIVE   0x20
00044 #define SPI_STATUS_SUCCESSFUL                   0x30
00045 #define SPI_STATUS_DELAY_US                     0x01
00046
00047 #define MC33996_DATA                            6
00048 #define MC33996_DATA_LEN                        8
00049
00050         /*
00051          * HIDAPI I/O structure
00052          */
00053         typedef struct {
00054                 hid_device *handle;
00055                 struct hid_device_info *devs, *cur_dev;
00056                 uint8_t buf[COMMAND_BUFFER_LENGTH]; // command buffer written to MCP2210
00057                 uint8_t rbuf[RESPONSE_BUFFER_LENGTH]; // response buffer
00058                 int32_t res; // # of bytes sent from hid_read(), hid_write() functions
00059         } mcp2210_spi_type;
00060
00061         void cbufs();
00062         int32_t SendUSBCmd(hid_device *, uint8_t *, uint8_t *);
00063         int32_t get_usb_res(void);
00064         void sleep_us(const uint32_t);
00065         bool get_MCP2210_ext_interrupt(void);
00066         int32_t cancel_spi_transfer(void);
00067         bool SPI_WriteRead(hid_device *, uint8_t *, uint8_t *);
00068         bool SPI_MCP2210_WriteRead(uint8_t* pTransmitData, const size_t txSize, uint8_t* pReceiveData,
      const size_t rxSize);
00069         void setup_tic12400_transfer(void);
00070         void get_tic12400_transfer(void);
00071         void setup_bmx160_transfer(uint8_t);
00072         void get_bmx160_transfer(void);
00073         void show_bmx160_transfer(void);
00074         void move_bmx160_transfer_data(uint8_t *);
00075         void move_bmx160_transfer_status(uint8_t *);
00076         uint8_t bmx160_get(uint8_t, uint8_t);
00077         uint8_t bmx160_set(uint8_t, uint8_t);
00078         uint8_t bmx160_set3(uint8_t *, uint8_t);
00079         void bmx160_update(void);
00080         void mc33996_init(void);
00081         bool mc33996_check(void);
00082         void mc33996_set(uint8_t, uint8_t, uint8_t);
00083         void setup_mc33996_transfer(uint8_t);
```

```
00084         void get_mc33996_transfer(void);
00085         void mc33996_update(void);
00086         mcp2210_spi_type* hidrawapi_mcp2210_init(const wchar_t *serial_number);
00087
00088
00089 #ifdef __cplusplus
00090 }
00091 #endif
00092
00093 #endif /* BMCDIO_H */
00094
```

4.10 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:
```

4.11 bmc.o.d

```
00001 build/Release/GNU-Linux/bmc.o: bmc.c daq.h bmc.h bmc_mqtt.h
00002 daq.h:
00003 bmc.h:
00004 bmc_mqtt.h:
```

4.12 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:
```

4.13 bmc_mqtt.o.d

```
00001 build/Release/GNU-Linux/bmc_mqtt.o: bmc_mqtt.c bmc_mqtt.h bmc.h daq.h
00002 bmc_mqtt.h:
00003 bmc.h:
00004 daq.h:
```

4.14 daq.o.d

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

4.15 daq.o.d

```
00001 build/Release/GNU-Linux/daq.o: daq.c daq.h bmc.h bmc_mqtt.h
00002 daq.h:
00003 bmc.h:
00004 bmc_mqtt.h:
```

4.16 daq.c

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

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

```
00171 {
00172     if (ADC_OPEN) {
00173         da_range->min = min_range;
00174         da_range->max = max_range;
00175         return 0;
00176     } else {
00177         return -1;
00178     }
00179 }
00180
00181 int set_dac_volts(int chan, double voltage)
00182 {
00183     lsampl_t data;
00184     int retval;
00185
00186     data = comedi_from_phys(voltage, da_range, maxdata_ao);
00187     bmc.dac_sample[chan] = data;
00188     retval = comedi_data_write(it, subdev_ao, chan, range_ao, aref_ao, data);
00189     if (retval < 0) {
00190         comedi_perror("comedi_data_write in set_dac_volts");
00191         DAC_ERROR = true;
00192     }
00193     return retval;
00194 }
00195
00196 int set_dac_raw(int chan, lsampl_t voltage)
00197 {
00198     int retval;
00199
00200     retval = comedi_data_write(it, subdev_ao, chan, range_ao, aref_ao, voltage);
00201     if (retval < 0) {
00202         comedi_perror("comedi_data_write in set_dac_raw");
00203         DAC_ERROR = true;
00204     }
00205     return retval;
00206 }
00207
00208 double get_adc_volts(int chan)
00209 {
00210     lsampl_t data[16];
00211     int retval;
00212
00213     retval = comedi_data_read_n(it, subdev_ai, chan, range_ai, aref_ai, &data[0], 1);
00214     if (retval < 0) {
00215         comedi_perror("comedi_data_read in get_adc_volts");
00216         ADC_ERROR = true;
00217         return 0.0;
00218     }
00219     bmc.adc_sample[chan] = data[0];
00220
00221     ad_range->min = 0.0f;
00222     if (bmc.BOARD == bmcboard) {
00223         if (chan == channel_ANA4 || chan == channel_ANA5) {
00224             if (chan == channel_ANA4) {
00225                 ad_range->max = HV_SCALE4;
00226             } else {
00227                 ad_range->max = HV_SCALE5;
00228             }
00229         } else {
00230             ad_range->max = HV_SCALE_RAW;
00231             ad_range->min = 0.0f;
00232         }
00233     } else {
00234         ad_range->max = ha_daq_host.scaler[ha_daq_host.hindex];
00235     }
00236
00237     return comedi_to_phys(data[0], ad_range, maxdata_ai);
00238 }
00239
00240 int set_dio_output(int chan)
00241 {
00242     return comedi_dio_config(it,
00243         subdev_dio,
00244         chan,
00245         COMEDI_OUTPUT);
00246 }
00247
00248 int set_dio_input(int chan)
00249 {
00250     return comedi_dio_config(it,
00251         subdev_dio,
00252         chan,
00253         COMEDI_INPUT);
00254 }
00255
00256 int get_dio_bit(int chan)
00257 {
```

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

```
00345         fprintf(fout, "Ranges %i \r\n", ranges_dio);
00346     }
00347
00348     if (PWM_OPEN) {
00349         fprintf(fout, "Subdev COU %i ", subdev_counter);
00350         channels_counter = comedi_get_n_channels(it, subdev_counter);
00351         fprintf(fout, "Digital Channels %i ", channels_counter);
00352         maxdata_counter = comedi_get_maxdata(it, subdev_counter, i);
00353         fprintf(fout, "Maxdata %i ", maxdata_counter);
00354         ranges_counter = comedi_get_n_ranges(it, subdev_counter, i);
00355         fprintf(fout, "Ranges %i \r\n", ranges_counter);
00356     }
00357     return 0;
00358 }
00359
00360 int get_data_sample(void)
00361 {
00362     unsigned int obits;
00363
00364     bmc.datain.D0 = get_dio_bit(0);
00365
00366     if (JUST_BITS) { // send I/O bit by bit
00367         put_dio_bit(0, bmc.dataout.d.D0);
00368         put_dio_bit(1, bmc.dataout.d.D1);
00369         put_dio_bit(2, bmc.dataout.d.D2);
00370         put_dio_bit(3, bmc.dataout.d.D3);
00371         put_dio_bit(4, bmc.dataout.d.D4);
00372         put_dio_bit(5, bmc.dataout.d.D5);
00373         put_dio_bit(6, bmc.dataout.d.D6);
00374         put_dio_bit(7, bmc.dataout.d.D7);
00375     } else { // send I/O as a byte mask
00376         obits = bmc.dataout.dio_buf;
00377         comedi_dio_bitfield2(it, subdev_do, 0xff, &obits, 0);
00378     }
00379
00380     return 0;
00381 }
00382
00383 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
00384 {
00385     static double smooth[LPCHANC] = {0};
00386     double lp_speed, lp_x;
00387
00388     if ((bn >= LPCHANC) || (bn < 0)) // check for proper array position
00389         return new;
00390     if (slow) {
00391         lp_speed = 0.033;
00392     } else {
00393         lp_speed = 0.125;
00394     }
00395     lp_x = ((smooth[bn]*100.0) + (((new * 100.0)-(smooth[bn]*100.0)) * lp_speed)) / 100.0;
00396     smooth[bn] = lp_x;
00397     if (slow == (-1)) { // reset and return zero
00398         lp_x = 0.0;
00399         smooth[bn] = 0.0;
00400     }
00401     return lp_x;
00402 }
```

4.17 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          * scale adc result into calibrated units
00031          * for USB boards, BMC boards use driver ranges
00032          */
00033 #define HV_SCALE0        83.6f
00034 #define HV_SCALE1        74.4f
00035 #define HV_SCALE2        83.6f
00036 #define HV_SCALE3        83.6f
00037 #define HV_SCALE4        83.6f
00038 #define HV_SCALE5        83.6f
00039 #define HV_SCALE_RAW    4.096f
00040
00041
00042 #define OVER_SAMP       1
00043
00044 #include <stdint.h>
00045 #include <comedilib.h>
00046 #include "bmc.h"
00047
00048 #define BMCBoard        "BMCBoard"
00049 #define USBBoard        "K8055 (VM110)"
00050
00051         typedef enum {
00052                 bmcboard = 0,
00053                 usbboard = 1,
00054         } board_t;
00055
00056         typedef enum {
00057                 channel_ANA0 = 0x0,
00058                 channel_ANA1 = 0x1,
00059                 channel_ANA2 = 0x2,
00060                 channel_ANA4 = 0x4,
00061                 channel_ANA5 = 0x5,
00062                 channel_ANC6 = 0x16,
00063                 channel_ANC7 = 0x17,
00064                 channel_AND5 = 0x1D,
00065                 channel_VSS = 0x3B,
00066                 channel_Temp = 0x3C,
00067                 channel_DAC1 = 0x3D,
00068                 channel_FVR_Buffer1 = 0x3E,
00069                 channel_FVR_Buffer2 = 0x3F
00070         } ADC_channel_t;
00071
00072         struct didata {
00073                 uint32_t D0 : 1; //
00074                 uint32_t D1 : 1; //
00075                 uint32_t D2 : 1; //
00076                 uint32_t D3 : 1; //
00077                 uint32_t D4 : 1; //
00078                 uint32_t D5 : 1; //
00079                 uint32_t D6 : 1; //
00080                 uint32_t D7 : 1; //
00081         };
00082
00083         union dio_buf_type {
00084                 uint32_t dio_buf;
00085                 struct didata d;
00086         };
00087
00088         typedef struct bmcdata {
00089                 double pv_voltage, cc_voltage, input_voltage, b1_voltage, b2_voltage, system_voltage,
      logic_voltage;
00090                 double pv_current, cc_current, battery_current;
00091                 struct didata datain;
00092                 union dio_buf_type dataout;
00093                 int32_t adc_sample[32];
00094                 int32_t dac_sample[32];
00095                 int32_t utc;
00096                 board_t BOARD;
00097                 char * BNAME;
00098         } bmctype;
00099
00100         extern volatile struct bmcdata bmc;
00101         extern struct didata datain;
00102         extern struct dodata dataout;
00103
00104         extern int maxdata_ai, ranges_ai, channels_ai;
00105         extern int maxdata_ao, ranges_ao, channels_ao;
00106         extern int maxdata_di, ranges_di, channels_di, datain_di;
00107         extern int maxdata_do, ranges_do, channels_do, datain_do;
```

```
00108         extern int maxdata_counter, ranges_counter, channels_counter, datain_counter;
00109
00110         int init_daq(double, double, int);
00111         int init_dac(double, double, int);
00112         int init_dio(void);
00113         int adc_range(double, double);
00114         int dac_range(double, double);
00115         double get_adc_volts(int);
00116         int set_dac_volts(int, double);
00117         int set_dac_raw(int, lsampl_t);
00118         int get_dio_bit(int);
00119         int put_dio_bit(int, int);
00120         int set_dio_input(int);
00121         int set_dio_output(int);
00122         int get_data_sample(void);
00123         double lp_filter(double, int, int);
00124 #ifdef __cplusplus
00125 }
00126 #endif
00127
00128 #endif /* DAQ_H */
00129
```

4.18 mc33996.c

```
00001 #include "bmcdio.h"
00002
00003 static const char *build_date = __DATE__, *build_time = __TIME__;
00004
00005 void mc33996_version(void)
00006 {
00007     printf("\r--- MC33996 Driver Version %s %s %s ---\r\n", MC33996_DRIVER, build_date, build_time);
00008 }
```

4.19 mc33996.h

```
00001 /*
00002  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
00003  * Click nbfs://nbhost/SystemFileSystem/Templates/cFiles/file.h to edit this template
00004  */
00005
00006 /*
00007  * File:   mc33996.h
00008  * Author: root
00009  *
00010  * Created on May 12, 2025, 11:11AM
00011  */
00012
00013 #ifndef MC33996_H
00014 #define MC33996_H
00015
00016 #ifdef __cplusplus
00017 extern "C" {
00018 #endif
00019
00020 #include <stdlib.h>
00021 #include <stdbool.h>
00022 #include <stdint.h>
00023
00024 #define MC33996_DRIVER "V0.3"
00025
00026         /*
00027          * MC33996 command structure
00028          */
00029         typedef struct __attribute__((packed))
00030         {
00031                 uint16_t out;
00032                 uint8_t cmd;
00033         }
00034         mc33996buf_type;
00035
00036         /*
00037          * MC33996 response structure
00038          */
00039         typedef struct __attribute__((packed))
00040         {
00041                 uint16_t out_faults;
00042                 uint8_t faults;
```

```
00043         }
00044         mc33996read_type;
00045
00046 #define mc33996_control         0x00
00047 #define mc33996_load            0x04
00048 #define mc33996_reset           0x18
00049 #define mc33996_magic_h         0x19
00050 #define mc33996_magic_l         0x57
00051
00052 #define MC33996_DATA            6
00053 #define MC33996_DATA_LEN        8
00054
00055         void mc33996_version(void);
00056
00057
00058 #ifdef __cplusplus
00059 }
00060 #endif
00061
00062 #endif /* MC33996_H */
00063
```

4.20 mcp2210.h

```
00001 /*
00002  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
00003  * Click nbfs://nbhost/SystemFileSystem/Templates/cFiles/file.h to edit this template
00004  */
00005
00006 /*
00007  * File:   mcp2210.h
00008  * Author: root
00009  *
00010  * Created on May 12, 2025, 11:09AM
00011  */
00012
00013 #ifndef MCP2210_H
00014 #define MCP2210_H
00015
00016 #ifdef __cplusplus
00017 extern "C" {
00018 #endif
00019
00020
00021
00022
00023 #ifdef __cplusplus
00024 }
00025 #endif
00026
00027 #endif /* MCP2210_H */
00028
```

4.21 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
```

4.22 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 <cstdarg>      // 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 utilites
00052 #include <cstddef>      // 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 <cfloat>       // 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 <clocale>          // 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 <typeindex>        // (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 utilites
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
```

4.23 tic12400.c

```
00001 /*
00002  * TC12400 driver for Q84 v0.1
00003  * uses SPI5 mode1 at 4MHz no interrupts
00004  * external interrupt 2 is used to detect chip switch events
00005  */
00006
00007 #include "tic12400.h"
00008 #include "bmc_mqtt.h"
00009
00010 /*
00011  * command structure data
00012  * the parity bit must be set correctly for the command to execute on the chip
00013  */
00014 const ticbuf_type setup32 = {
00015     .wr = 1,
00016     .addr = 0x32,
00017     .data = 0,
00018     .par = 1,
00019 };
00020 const ticbuf_type setup21 = {
00021     .wr = 1,
00022     .addr = 0x21,
00023     .data = 0,
00024     .par = 0,
00025 };
00026 const ticbuf_type setup1c = {
00027     .wr = 1,
00028     .addr = 0x1c,
00029     .data = 0,
00030     .par = 1,
00031 };
00032 const ticbuf_type setup1b = {
00033     .wr = 1,
00034     .addr = 0x1b,
00035     .data = 0xffffff,
00036     .par = 0,
00037 };
00038 const ticbuf_type setup1a = {
00039     .wr = 1,
00040     .addr = 0x1a,
00041     .data = 0xc000,
00042     .par = 1,
00043 };
```

```
00044 const ticbuf_type setup1a_trigger = {
00045     .wr = 1,
00046     .addr = 0x1a,
00047     .data = 0x0a00, // trigger and do config register CRC
00048     .par = 1,
00049 };
00050 const ticbuf_type setup22 = {
00051     .wr = 1,
00052     .addr = 0x22,
00053     .data = 0xffffff,
00054     .par = 0,
00055 };
00056 const ticbuf_type setup23 = {
00057     .wr = 1,
00058     .addr = 0x23,
00059     .data = 0xffffff,
00060     .par = 1,
00061 };
00062 const ticbuf_type setup24 = {
00063     .wr = 1,
00064     .addr = 0x24,
00065     .data = 0xfff,
00066     .par = 0,
00067 };
00068 const ticbuf_type setup1d = {
00069     .wr = 1,
00070     .addr = 0x1d,
00071     .data = 011111111, // octal constant, all inputs 1mA wetting current
00072     .par = 0,
00073 };
00074 const ticbuf_type ticread05 = {
00075     .wr = 0,
00076     .addr = 0x05,
00077     .data = 0,
00078     .par = 1,
00079 };
00080 const ticbuf_type ticdevid01 = {
00081     .wr = 0,
00082     .addr = 0x01,
00083     .data = 0,
00084     .par = 0,
00085 };
00086 const ticbuf_type ticstat02 = {
00087     .wr = 0,
00088     .addr = 0x02,
00089     .data = 0,
00090     .par = 0,
00091 };
00092 const ticbuf_type ticreset1a = {
00093     .wr = 1,
00094     .addr = 0x1a,
00095     .data = 0x1,
00096     .par = 0,
00097 };
00098
00099 /*
00100  * global status and value registers
00101  */
00102 volatile uint32_t tic12400_status = 0, tic12400_counts = 0, tic12400_value_counts = 0;
00103 volatile uint32_t tic12400_value = 0;
00104 volatile bool tic12400_init_fail = false, tic12400_event = false; // chip error detection flag
00105 volatile bool tic12400_parity_status = false;
00106 volatile int32_t tic12400_fail_value = 0;
00107
00108 static ticread_type *ticstatus = (ticread_type*) & tic12400_status;
00109 static ticread_type *ticvalue = (ticread_type*) & tic12400_value;
00110
00111 static const char *build_date = __DATE__, *build_time = __TIME__;
00112
00113 void tic12400_version(void)
00114 {
00115     printf("\r--- TIC12400 Driver Version %s %s %s ---\r\n", TIC12400_DRIVER, build_date, build_time);
00116 }
00117 /*
00118  * software reset of the chip using SPI
00119  * all registers set to their default values
00120  */
00121 /*
00122  * chip setup via SPI data transfers
00123  */
00124 void tic12400_reset(void)
00125 {
00126     tic12400_wr(&ticreset1a, 4);
00127     tic12400_wr(&ticreset1a, 4);
00128     tic12400_fail_value = 1;
00129 }
00130
```

```
00131 /*
00132  * chip detection and configuration for all inputs with interrupts for
00133  * switch state changes with debounce
00134  * returns false on configuration failure
00135  */
00136
00137 /*
00138  * chip setup via SPI data transfers
00139  */
00140 bool tic12400_init(void)
00141 {
00142     tic12400_status = tic12400_wr(&ticstat02, 0); // get status to check for proper operation
00143
00144     if ((ticstatus->data > por_bit) || !ticstatus->por) { // check for any high bits beyond POR bits
      set
00145         tic12400_init_fail = true;
00146         tic12400_fail_value = -1;
00147         goto fail;
00148     }
00149
00150     tic12400_wr(&setup32, 0); //all set to compare mode, 0x32
00151     tic12400_wr(&setup21, 0); //Compare threshold all bits 2V, 0x21
00152     tic12400_wr(&setup1c, 0); //all set to GND switch type, 0x1c
00153     tic12400_wr(&setup1b, 0); //all channels are enabled, 0x1b
00154     tic12400_wr(&setup22, 0); //set switch interrupts, 0x22
00155     tic12400_wr(&setup23, 0); //set switch interrupts, 0x23
00156     tic12400_wr(&setup24, 0); // enable interrupts, 0x24
00157     tic12400_wr(&setup1d, 0); // set wetting currents, 0x1d
00158     tic12400_wr(&setup1a, 0); // set switch debounce to max 4 counts, 0x1a
00159     tic12400_status = tic12400_wr(&setup1a_trigger, 2); // trigger switch detections & CRC, 0x1a
00160
00161     if (ticstatus->spi_fail) {
00162         tic12400_init_fail = true;
00163         tic12400_fail_value = -2;
00164         goto fail;
00165     }
00166
00167     tic12400_status = tic12400_wr(&ticdevid01, 0); // get device id, 0x01
00168
00169 fail:
00170     return !tic12400_init_fail; // flip to return true if NO configuration failures
00171 }
00172
00173 /*
00174  * send tic12400 commands to SPI port 5 with possible delay after transfer
00175  * returns 32-bit spi response from the tic12400
00176  */
00177 uint32_t tic12400_wr(const ticbuf_type * buffer, uint16_t del)
00178 {
00179     static uint32_t rbuffer = 0;
00180
00181     SPI_MCP2210_WriteRead((void*) buffer, 4, (void*) &rbuffer, 4);
00182
00183     if (ticvalue->parity_fail) { // check for command parity errors
00184         tic12400_parity_status = true;
00185     };
00186
00187     if (del) {
00188         sleep_us(del * 1000);
00189     }
00190
00191     return rbuffer;
00192 }
00193
00194 /*
00195  * switch data reading testing routine
00196  * tic12400 value is updated in external interrupt #2 ISR
00197  */
00198 uint32_t tic12400_get_sw(void)
00199 {
00200     if (tic12400_init_fail) { // Trouble in River City
00201         return 0;
00202     }
00203
00204     if (tic12400_value & (raw_mask_0)) {
00205         //      BSP_LED1_Clear();
00206     } else {
00207         //      BSP_LED1_Set();
00208     }
00209
00210     if (tic12400_value & (raw_mask_11)) {
00211         //      BSP_LED2_Clear();
00212     } else {
00213         //      BSP_LED2_Set();
00214     }
00215
00216     tic12400_event = false;
```

```
00217     return tic12400_value;
00218 }
00219
00220 /*
00221  * 32-bit 1's parity check
00222  * https://graphics.stanford.edu/~seander/bithacks.html#ParityNaive
00223  */
00224 bool tic12400_parity(uint32_t v)
00225 {
00226     v ^= v » 16;
00227     v ^= v » 8;
00228     v ^= v » 4;
00229     v &= 0xf;
00230     return(0x6996 » v) & 1;
00231 }
00232
00233 /*
00234  * switch SPI status and switch data updates
00235  * sets event flag for user application notification
00236  */
00237 void tic12400_read_sw(uint32_t a, uintptr_t b)
00238 {
00239     tic12400_value = tic12400_wr(&ticread05, 0); // read switch
00240     tic12400_status = tic12400_wr(&ticstat02, 0); // read status
00241
00242     if (ticvalue->ssc && tic12400_parity(tic12400_value)) { // only trigger on switch state change
00243         tic12400_event = true;
00244         tic12400_value_counts++;
00245     }
00246     tic12400_counts++;
00247 }
00248
```

4.24 tic12400.h

```
00001 /*
00002  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
00003  * Click nbfs://nbhost/SystemFileSystem/Templates/cFiles/file.h to edit this template
00004  */
00005
00006 /*
00007  * File:   tic12400.h
00008  * Author: root
00009  *
00010  * Created on May 12, 2025, 11:11AM
00011  */
00012
00013 #ifndef TIC12400_H
00014 #define TIC12400_H
00015
00016 #ifdef __cplusplus
00017 extern "C" {
00018 #endif
00019
00020 #include <stdlib.h>
00021 #include <stdbool.h>
00022 #include <stdint.h>
00023
00024 #define TIC12400_DRIVER "V0.5"
00025
00026 #define por_bit 0x01
00027 /*
00028  * switch bit masks in the raw 32-bit register from the TIC12400
00029  */
00030 #define raw_mask_0 0b010
00031 #define raw_mask_11 0b100000000000000
00032
00033 /*
00034  * TIC12400 command structure
00035  */
00036 typedef struct __attribute__((packed))
00037 {
00038     uint32_t par : 1;
00039     uint32_t data : 24;
00040     uint32_t addr : 6;
00041     uint32_t wr : 1;
00042 }
00043 ticbuf_type;
00044
00045 /*
00046  * TIC12400 response structure
00047  */
00048 typedef struct __attribute__((packed))
```

```
00049 {
00050     uint32_t par : 1;
00051     uint32_t data : 24;
00052     uint32_t oi : 1;
00053     uint32_t temp : 1;
00054     uint32_t vs_th : 1;
00055     uint32_t ssc : 1;
00056     uint32_t parity_fail : 1;
00057     uint32_t spi_fail : 1;
00058     uint32_t por : 1;
00059 }
00060 ticread_type;
00061
00062 void tic12400_version(void);
00063 void tic12400_reset(void);
00064 bool tic12400_init(void);
00065 uint32_t tic12400_wr(const ticbuf_type *, uint16_t);
00066 uint32_t tic12400_get_sw(void);
00067 void tic12400_read_sw(uint32_t, uintptr_t);
00068 bool tic12400_parity(uint32_t);
00069
00070 extern volatile uint32_t tic12400_status, tic12400_counts, tic12400_value_counts;
00071 extern volatile uint32_t tic12400_value;
00072 extern volatile bool tic12400_init_fail, tic12400_event;
00073 extern volatile bool tic12400_parity_status;
00074 extern volatile int32_t tic12400_fail_value;
00075
00076
00077 #ifdef __cplusplus
00078 }
00079 #endif
00080
00081 #endif /* TIC12400_H */
00082
```