```
/** @file
    Revolt ZX-7717-675 433Mhz Power Meter.

    Copyright (C) 2024 Christian W. Zuckschwerdt <zany@triq.net
    * thx Christian for help!
      logs and code by Boing (dhs_mobil@google.com)
    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.
*/
/**
Revolt ZX-7717-675 433Mhz Power Meter. for Revolt ZX-7716 Monitor.
* other names: HPM-27717, ZX-7717-919
* up to 6 channels
* first seen: 12-2024
* https://www.revolt-power.de/TOP-KAT161-Zusaetzliche-Steckdose-ZX-7717-
919.shtml
*
Outputs are: (in this order)
* Current (A) max 15.999 A , Minimum is >= 0.001 A
* Voltage (V) max 250.0 V
* Power  (VA) max 3679.9 VA
* PF (Powerfactor not in message, but calculated
* 8 bit checksum
* + some yet unknown bytes/flags
*
* HF Output is 10mW, but seems much higher (mb a very good antenna inside)
* RSSI : -0.1 dB SNR: 33.0 dB Noise: -33.1 dB at 31 mtr distance !
* Modulation: ASK/ OOK_PULSE_MANCHESTER_ZEROBIT
* send intervall: 5 sek and/or bycurrent change
*
The packet is 14 manchester encoded bytes with a Preamble of 0x2A and
a 8-bit checksum (last byte)
* some data:
*
* 2024-12-25 20:30:05
model     : Revolt ZX-7717-675                    id        : d507
Channel   : 6
UnknownB3 : 5aa8         UnknownB5 : 065c         Current   : 0.839 A
Voltage   : 227.9 V
Power     : 110.7 W      data      : 0dd5075aa8065c4703e7085304dd
Integrity : CRC
Modulation: ASK          Freq      : 434.0 MHz
RSSI      : -0.1 dB      SNR       : 34.3 dB       Noise     : -34.4 dB
*
* 0dd5075aa8065c2800ea08210088
* 0dd5075aa8065cc800ea08fb0002
* 0dd5075aa8065c4703e7085304dd
* 0dd5075aa8065c5803e70883041e
* 0dd5075aa8065cbe00de08f200e3
*
* 0d  : int8      payload_length (13)
* d507      : int16     id
* 5aa8      : int16     unknown1
* 06  : int8      channel (6) // TODO
* 5c  ; int8  unknown2
* be00      : int16     current (0.190)
* de08      : int16 voltage   (227.0)
* f200      : int16 power     (24.2)
* e3  : int8      checksum
*
*/
```

```
#include "decoder.h"

// Function definition
unsigned char reflect(unsigned char b);

unsigned char reflect(unsigned char b) {
   b = (b & 0xF0) >> 4 | (b & 0x0F) << 4;
   b = (b & 0xCC) >> 2 | (b & 0x33) << 2;
   b = (b & 0xAA) >> 1 | (b & 0x55) << 1;
   return b;
}

static int revolt_zx7717_decode(r_device *decoder, bitbuffer_t *bitbuffer)
{
    data_t *data;
    uint8_t search = 0x2a; // preamble is 0x2a
    uint8_t msg[42];
    uint8_t msg_reflect[42];
    unsigned msg_len = 0;
    unsigned len;
    unsigned pos;
    unsigned i;
    int id;
    uint16_t channel;
    uint16_t current;
    uint16_t voltage;
    uint16_t power;
    uint32_t pf = 0;
    double powerf;
    uint16_t unknown1 = 0;
    uint16_t unknown2 = 0;

    int crc;
    int crc_calculated;
    char code_str[42];

      if (bitbuffer->num_rows != 1) {
        return DECODE_ABORT_EARLY;
    }
      msg_len = bitbuffer->bits_per_row[0];
          if (msg_len < 120 || msg_len > 120) { // TODO
            return DECODE_ABORT_EARLY; // Unrecognized data
    }

    pos = bitbuffer_search(bitbuffer, 0, 0, &search, 8);

    if (pos > 8) {
            return DECODE_ABORT_LENGTH; // short buffer or preamble not found
    }
    pos += 8; // skip preamble
    len = bitbuffer->bits_per_row[0] - pos;

    // we want 14 bytes (112 bits)
    if (len < 112) {
            return DECODE_ABORT_LENGTH; // short buffer
    }
    len = 112; // cut the last pulse

    bitbuffer_extract_bytes(bitbuffer, 0, pos, (uint8_t *)&msg, len);

      for (i=0; i<(len+7)/8 ; ++i) {
            msg_reflect[i] = reflect (msg[i]);
      }
```

```
      crc =  msg_reflect[13];
      crc_calculated = 0;

      for (i=0; i<12 ; ++i) {
            crc_calculated = crc_calculated + msg_reflect[i];
    }
    if (crc != ( crc_calculated & 255)) {
            return DECODE_FAIL_MIC; // bad crc
      }

      id = (msg_reflect[1] << 8) | (msg_reflect[2]);
      unknown1 = (msg_reflect[3] << 8) | msg_reflect[4];
      unknown2 = (msg_reflect[5] << 8) | msg_reflect[6];
      channel = (msg_reflect[5] ); // not shure yet
      current = (msg_reflect[8]<< 8) | msg_reflect[7];
      voltage = (msg_reflect[10]<< 8) | msg_reflect[9];
      power = (msg_reflect[12] << 8 ) | msg_reflect[11];
      // calculation for PF (Powerfactor) fails if current is <  0.02 A
      // most batterycharger will fail in standby
      powerf = power / (current * (voltage * 0.001));

    for (i=0; i<(len+7)/8 ; ++i) {
            sprintf(&code_str[i*2], "%02x", msg_reflect[i]);
    }

    /* clang-format off */
    data = data_make(
            "model",    "",             DATA_STRING,    "Revolt ZX-7717-675",
            "device_no",      "id",       DATA_FORMAT, "%04x", DATA_INT, id,
            "channel",        "Channel",  DATA_INT, channel,
            "unknown",        "UnknownB3",  DATA_FORMAT, "%04x", DATA_INT,
unknown1,
            "unknown",        "UnknownB5",  DATA_FORMAT, "%04x", DATA_INT,
unknown2,
            "current_A",      "Current",  DATA_FORMAT, "%.3f A", DATA_DOUBLE,
current* 0.001,
            "voltage_V",      "Voltage", DATA_FORMAT, "%.1f V", DATA_DOUBLE,
voltage * 0.1,
                  "power",          "Power",    DATA_FORMAT, "%.1f W",
      DATA_DOUBLE, power * 0.1,
            "pf",             "PF(calc)",       DATA_DOUBLE, powerf,
            "code",     "data",    DATA_STRING,    code_str,
            "mic",      "Integrity",    DATA_STRING,    "CRC",
            NULL);
    /* clang-format on */

    decoder_output_data(decoder, data);
    return 1;
}

static char const *const output_fields[] = {
        "model",
        "id",
        "channel",
        "unknown1",
        "unknown2",
        "current",
        "voltage",
        "power",
        "pf",
        "code",
        "mic",
        NULL,
};
```

```
r_device const revolt_zx7717 = {
          .name        = "Revolt ZX-7717 Enery Meter",
        .modulation  = OOK_PULSE_MANCHESTER_ZEROBIT,
        .short_width = 310, // Nominal width of clock half period [us]
        .long_width  = 310,
        .reset_limit = 900, // Maximum gap size before End Of Message [us]
        .decode_fn   = &revolt_zx7717_decode,
        .fields      = output_fields,
};
```