# Kernel driver pmbus

Supported chips:

- Flex BMR310, BMR453, BMR454, BMR456, BMR457, BMR458, BMR480, BMR490, BMR491, BMR492

  Prefixes: 'bmr310', 'bmr453', 'bmr454', 'bmr456', 'bmr457', 'bmr458', 'bmr480', 'bmr490', 'bmr491', 'bmr492'

  Addresses scanned: -

  Datasheets:

  https://flexpowermodules.com/products

- ON Semiconductor ADP4000, NCP4200, NCP4208

  Prefixes: 'adp4000', 'ncp4200', 'ncp4208'

  Addresses scanned: -

  Datasheets:

  https://www.onsemi.com/pub_link/Collateral/ADP4000-D.PDF

  https://www.onsemi.com/pub_link/Collateral/NCP4200-D.PDF

  https://www.onsemi.com/pub_link/Collateral/JUNE%202009-%20REV.%200.PDF

- Lineage Power

  Prefixes: 'mdt040', 'pdt003', 'pdt006', 'pdt012', 'udt020'

  Addresses scanned: -

  Datasheets:

  http://www.lineagepower.com/oem/pdf/PDT003A0X.pdf

  http://www.lineagepower.com/oem/pdf/PDT006A0X.pdf

  http://www.lineagepower.com/oem/pdf/PDT012A0X.pdf

  http://www.lineagepower.com/oem/pdf/UDT020A0X.pdf

  http://www.lineagepower.com/oem/pdf/MDT040A0X.pdf

- Texas Instruments TPS40400, TPS544B20, TPS544B25, TPS544C20, TPS544C25

  Prefixes: 'tps40400', 'tps544b20', 'tps544b25', 'tps544c20', 'tps544c25'

  Addresses scanned: -

  Datasheets:

  https://www.ti.com/lit/gpn/tps40400

  https://www.ti.com/lit/gpn/tps544b20

  https://www.ti.com/lit/gpn/tps544b25

  https://www.ti.com/lit/gpn/tps544c20

  https://www.ti.com/lit/gpn/tps544c25

- Maxim MAX20796

  Prefix: 'max20796'

  Addresses scanned: -

  Datasheet:

  Not published

- Generic PMBus devices

  Prefix: 'pmbus'

  Addresses scanned: -

  Datasheet: n.a.

Author: Guenter Roeck <linux@roeck-us.net>

## Description

This driver supports hardware monitoring for various PMBus compliant devices. It supports voltage, current, power, and temperature sensors as supported by the device.

Each monitored channel has its own high and low limits, plus a critical limit.

Fan support will be added in a later version of this driver.

## Usage Notes

This driver does not probe for PMBus devices, since there is no register which can be safely used to identify the chip (The MFG_ID register is not supported by all chips), and since there is no well defined address range for PMBus devices. You will have to instantiate the devices explicitly.

Example: the following will load the driver for an LTC2978 at address 0x60 on I2C bus #1:

```
$ modprobe pmbus
$ echo ltc2978 0x60 > /sys/bus/i2c/devices/i2c-1/new_device
```

## Platform data support

Support for additional PMBus chips can be added by defining chip parameters in a new chip specific driver file. For example, (untested) code to add support for Emerson DS1200 power modules might look as follows:

```
static struct pmbus_driver_info ds1200_info = {
      .pages = 1,
      /* Note: All other sensors are in linear mode */
      .direct[PSC_VOLTAGE_OUT] = true,
      .direct[PSC_TEMPERATURE] = true,
      .direct[PSC_CURRENT_OUT] = true,
      .m[PSC_VOLTAGE_IN] = 1,
      .b[PSC_VOLTAGE_IN] = 0,
      .R[PSC_VOLTAGE_IN] = 3,
      .m[PSC_VOLTAGE_OUT] = 1,
      .b[PSC_VOLTAGE_OUT] = 0,
      .R[PSC_VOLTAGE_OUT] = 3,
      .m[PSC_TEMPERATURE] = 1,
      .b[PSC_TEMPERATURE] = 0,
      .R[PSC_TEMPERATURE] = 3,
      .func[0] = PMBUS_HAVE_VIN | PMBUS_HAVE_IIN | PMBUS_HAVE_STATUS_INPUT
                 | PMBUS_HAVE_VOUT | PMBUS_HAVE_STATUS_VOUT
                 | PMBUS_HAVE_IOUT | PMBUS_HAVE_STATUS_IOUT
                 | PMBUS_HAVE_PIN | PMBUS_HAVE_POUT
                 | PMBUS_HAVE_TEMP | PMBUS_HAVE_STATUS_TEMP
                 | PMBUS_HAVE_FAN12 | PMBUS_HAVE_STATUS_FAN12,
};

static int ds1200_probe(struct i2c_client *client)
{
      return pmbus_do_probe(client, &ds1200_info);
}

static const struct i2c_device_id ds1200_id[] = {
      {"ds1200", 0},
      {}
};

MODULE_DEVICE_TABLE(i2c, ds1200_id);

/* This is the driver that will be inserted */
static struct i2c_driver ds1200_driver = {
      .driver = {
                 .name = "ds1200",
                 },
      .probe_new = ds1200_probe,
      .id_table = ds1200_id,
};

static int __init ds1200_init(void)
{
      return i2c_add_driver(&ds1200_driver);
}

static void __exit ds1200_exit(void)
{
```

```
        i2c_del_driver(&ds1200_driver);
}
```

## Sysfs entries

When probing the chip, the driver identifies which PMBus registers are supported, and determines available sensors from this information. Attribute files only exist if respective sensors are supported by the chip. Labels are provided to inform the user about the sensor associated with a given sysfs entry.

The following attributes are supported. Limits are read-write; all other attributes are read-only.

| | |
|---|---|
| inX_input | Measured voltage. From READ_VIN or READ_VOUT register. |
| inX_min | Minimum Voltage. From VIN_UV_WARN_LIMIT or VOUT_UV_WARN_LIMIT register. |
| inX_max | Maximum voltage. From VIN_OV_WARN_LIMIT or VOUT_OV_WARN_LIMIT register. |
| inX_lcrit | Critical minimum Voltage. From VIN_UV_FAULT_LIMIT or VOUT_UV_FAULT_LIMIT register. |
| inX_crit | Critical maximum voltage. From VIN_OV_FAULT_LIMIT or VOUT_OV_FAULT_LIMIT register. |
| inX_min_alarm | Voltage low alarm. From VOLTAGE_UV_WARNING status. |
| inX_max_alarm | Voltage high alarm. From VOLTAGE_OV_WARNING status. |
| inX_lcrit_alarm | Voltage critical low alarm. From VOLTAGE_UV_FAULT status. |
| inX_crit_alarm | Voltage critical high alarm. From VOLTAGE_OV_FAULT status. |
| inX_label | "vin", "vcap", or "voutY" |
| inX_rated_min | Minimum rated voltage. From MFR_VIN_MIN or MFR_VOUT_MIN register. |
| inX_rated_max | Maximum rated voltage. From MFR_VIN_MAX or MFR_VOUT_MAX register. |
| currX_input | Measured current. From READ_IIN or READ_IOUT register. |
| currX_max | Maximum current. From IIN_OC_WARN_LIMIT or IOUT_OC_WARN_LIMIT register. |
| currX_lcrit | Critical minimum output current. From IOUT_UC_FAULT_LIMIT register. |
| currX_crit | Critical maximum current. From IIN_OC_FAULT_LIMIT or IOUT_OC_FAULT_LIMIT register. |
| currX_alarm | Current high alarm. From IIN_OC_WARNING or IOUT_OC_WARNING status. |
| currX_max_alarm | Current high alarm. From IIN_OC_WARN_LIMIT or IOUT_OC_WARN_LIMIT status. |
| currX_lcrit_alarm | Output current critical low alarm. From IOUT_UC_FAULT status. |
| currX_crit_alarm | Current critical high alarm. From IIN_OC_FAULT or IOUT_OC_FAULT status. |
| currX_label | "iin", "iinY", "iinY.Z", "ioutY", or "ioutY.Z", where Y reflects the page number and Z reflects the phase. |
| currX_rated_max | Maximum rated current. From MFR_IIN_MAX or MFR_IOUT_MAX register. |
| powerX_input | Measured power. From READ_PIN or READ_POUT register. |
| powerX_cap | Output power cap. From POUT_MAX register. |
| powerX_max | Power limit. From PIN_OP_WARN_LIMIT or POUT_OP_WARN_LIMIT register. |
| powerX_crit | Critical output power limit. From POUT_OP_FAULT_LIMIT register. |
| powerX_alarm | Power high alarm. From PIN_OP_WARNING or POUT_OP_WARNING status. |
| powerX_crit_alarm | Output power critical high alarm. From POUT_OP_FAULT status. |
| powerX_label | "pin", "pinY", "pinY.Z", "poutY", or "poutY.Z", where Y reflects the page number and Z reflects the phase. |
| powerX_rated_max | Maximum rated power. From MFR_PIN_MAX or MFR_POUT_MAX register. |
| tempX_input | Measured temperature. From READ_TEMPERATURE_X register. |
| tempX_min | Minimum temperature. From UT_WARN_LIMIT register. |
| tempX_max | Maximum temperature. From OT_WARN_LIMIT register. |
| tempX_lcrit | Critical low temperature. From UT_FAULT_LIMIT register. |
| tempX_crit | Critical high temperature. From OT_FAULT_LIMIT register. |
| tempX_min_alarm | Chip temperature low alarm. Set by comparing READ_TEMPERATURE_X with UT_WARN_LIMIT if TEMP_UT_WARNING status is set. |
| tempX_max_alarm | Chip temperature high alarm. Set by comparing READ_TEMPERATURE_X with OT_WARN_LIMIT if TEMP_OT_WARNING status is set. |
| tempX_lcrit_alarm | Chip temperature critical low alarm. Set by comparing READ_TEMPERATURE_X with UT_FAULT_LIMIT if TEMP_UT_FAULT status is set. |
| tempX_crit_alarm | Chip temperature critical high alarm. Set by comparing READ_TEMPERATURE_X with OT_FAULT_LIMIT if TEMP_OT_FAULT status is set. |
| tempX_rated_min | Minimum rated temperature. From MFR_TAMBIENT_MIN register. |
| tempX_rated_max | Maximum rated temperature. From MFR_TAMBIENT_MAX, MFR_MAX_TEMP_1, MFR_MAX_TEMP_2 or MFR_MAX_TEMP_3 register. |