Kernel driver max6875

Supported chips:

- Maxim MAX6874, MAX6875

  Prefix: 'max6875'

  Addresses scanned: None (see below)

  Datasheet: http://pdfserv.maxim-ic.com/en/ds/MAX6874-MAX6875.pdf

Author: Ben Gardner <bgardner@wabtec.com>

Description

The Maxim MAX6875 is an EEPROM-programmable power-supply sequencer/supervisor. It provides timed outputs that can be used as a watchdog, if properly wired. It also provides 512 bytes of user EEPROM.

At reset, the MAX6875 reads the configuration EEPROM into its configuration registers. The chip then begins to operate according to the values in the registers.

The Maxim MAX6874 is a similar, mostly compatible device, with more inputs and outputs:

•	vin	gpi	vout
MAX6874	6	4	8
MAX6875	4	3	5

See the datasheet for more information.

Sysfs entries

eeprom - 512 bytes of user-defined EEPROM space.

General Remarks

Valid addresses for the MAX6875 are 0x50 and 0x52.

Valid addresses for the MAX6874 are 0x50, 0x52, 0x54 and 0x56.

The driver does not probe any address, so you explicitly instantiate the devices.

Example:

```
$ modprobe max6875
$ echo max6875 0x50 > /sys/bus/i2c/devices/i2c-0/new_device
```

The MAX6874/MAX6875 ignores address bit 0, so this driver attaches to multiple addresses. For example, for address 0x50, it also reserves 0x51. The even-address instance is called 'max6875', the odd one is 'dummy'.

Programming the chip using i2c-dev

Use the i2c-dev interface to access and program the chips.

Reads and writes are performed differently depending on the address range.

The configuration registers are at addresses 0x00 - 0x45.

Use i2c_smbus_write_byte_data() to write a register and i2c_smbus_read_byte_data() to read a register.

The command is the register number.

Examples:

To write a 1 to register 0x45:

```
i2c_smbus_write_byte_data(fd, 0x45, 1);
```

To read register 0x45:

```
value = i2c_smbus_read_byte_data(fd, 0x45);
```

The configuration EEPROM is at addresses 0x8000 - 0x8045.

The user EEPROM is at addresses 0x8100 - 0x82ff.

Use i2c_smbus_write_word_data() to write a byte to EEPROM.

The command is the upper byte of the address: 0x80, 0x81, or 0x82. The data word is the lower part of the address or'd with data << 8:

```
cmd = address >> 8;
val = (address & 0xff) | (data << 8);
```

Example:

To write 0x5a to address 0x8003:

```
i2c_smbus_write_word_data(fd, 0x80, 0x5a03);
```

Reading data from the EEPROM is a little more complicated.

Use i2c_smbus_write_byte_data() to set the read address and then i2c_smbus_read_byte() or i2c_smbus_read_i2c_block_data() to read the data.

Example:

To read data starting at offset 0x8100, first set the address:

```
i2c_smbus_write_byte_data(fd, 0x81, 0x00);
```

And then read the data:

```
value = i2c_smbus_read_byte(fd);
```

or:

```
count = i2c_smbus_read_i2c_block_data(fd, 0x84, 16, buffer);
```

The block read should read 16 bytes.

0x84 is the block read command.

See the datasheet for more details.