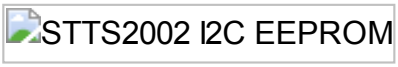
Linux中已经支持了大部分I2C EEPROM的read / write.

drivers/misc/eeprom/at24.c

> at24.c - handle most I2C EEPROMs

Gemstone2 TOC board上的两块I2C EEPROM可以被该driver support.
M24C64 I2C EEPROM


STTS2002 I2C EEPROM



```
static const struct i2c_device_id at24_ids[] = {
    /* needs 8 addresses as A0-A2 are ignored */
    { "24c00", AT24_DEVICE_MAGIC(128 / 8, AT24_FLAG_TAKE8ADDR) },
    /* old variants can't be handled with this generic entry! */
    { "24c01", AT24_DEVICE_MAGIC(1024 / 8, 0) },
    { "24c02", AT24_DEVICE_MAGIC(2048 / 8, 0) },
    /* spd is a 24c02 in memory DIMMs */
    { "spd", AT24_DEVICE_MAGIC(2048 / 8,
        AT24_FLAG_READONLY | AT24_FLAG_IRUGO) },
    { "24c04", AT24_DEVICE_MAGIC(4096 / 8, 0) },
    /* 24rf08 quirk is handled at i2c-core */
    { "24c08", AT24_DEVICE_MAGIC(8192 / 8, 0) },
    { "24c16", AT24_DEVICE_MAGIC(16384 / 8, 0) },
    { "24c32", AT24_DEVICE_MAGIC(32768 / 8, AT24_FLAG_ADDR16) },
    { "24c64", AT24_DEVICE_MAGIC(65536 / 8, AT24_FLAG_ADDR16) },
    { "24c128", AT24_DEVICE_MAGIC(131072 / 8, AT24_FLAG_ADDR16) },
    { "24c256", AT24_DEVICE_MAGIC(262144 / 8, AT24_FLAG_ADDR16) },
    { "24c512", AT24_DEVICE_MAGIC(524288 / 8, AT24_FLAG_ADDR16) },
    { "24c1024", AT24_DEVICE_MAGIC(1048576 / 8, AT24_FLAG_ADDR16) },
    { "at24", 0 },
    { /* END OF LIST */ }
};
MODULE_DEVICE_TABLE(i2c, at24_ids);
```

这两块EEPROM分别对应上表中的"24c64" (64 Kbit, 8 KByte)和"spd" (2Kb and 256 Byte)

这里的关键是要为这两个EEPROM device创建I2C device,完成at24 driver与i2c device的"match"。

---

in drivers/i2c/i2c-core.c

```
struct bus_type i2c_bus_type = {
    .name       = "i2c",
    .match      = i2c_device_match,
    .probe      = i2c_device_probe,
    .remove     = i2c_device_remove,
    .shutdown   = i2c_device_shutdown,
    .pm     = &i2c_device_pm_ops,
};
```

```
static int i2c_device_match(struct device *dev, struct device_driver *drv)
{
    struct i2c_client   *client = i2c_verify_client(dev);
    struct i2c_driver   *driver;

    if (!client)
        return 0;

    /* Attempt an OF style match */
    if (of_driver_match_device(dev, drv))
        return 1;

    /* Then ACPI style match */
    if (acpi_driver_match_device(dev, drv))
        return 1;

    driver = to_i2c_driver(drv);
    /* match on an id table if there is one */
    if (driver->id_table)
        return i2c_match_id(driver->id_table, client) != NULL;

    return 0;
}
```

i2c bus上完成driver和device配对的是 `i2c_device_match()` 。

由于at24.c没有支持device tree，所以使用的是driver中的 `id_table` 来match i2c device.

```
    /* match on an id table if there is one */
    if (driver->id_table)
        return i2c_match_id(driver->id_table, client) != NULL;
```

```
static const struct i2c_device_id *i2c_match_id(const struct i2c_device_id *
id,
                        const struct i2c_client *client)
{
    while (id->name[0]) {
        if (strcmp(client->name, id->name) == 0)
            return id;
        id++;
    }
    return NULL;
}
```

代表i2c device的i2c_client的 `name` field必须与at24.c中的 `id_table` 中的成员相等。即client->name为"24c64"或"spd"。

> struct i2c_client - represent an I2C slave device

in drivers/i2c/i2c-boardinfo.c

```
/**
 * i2c_register_board_info - statically declare I2C devices
 * @busnum: identifies the bus to which these devices belong
 * @info: vector of i2c device descriptors
 * @len: how many descriptors in the vector; may be zero to reserve
 *  the specified bus number.
 *
 * Systems using the Linux I2C driver stack can declare tables of board info
 * while they initialize.  This should be done in board-specific init code
 * near arch_initcall() time, or equivalent, before any I2C adapter driver is
 * registered.  For example, mainboard init code could define several devices,
 * as could the init code for each daughtercard in a board stack.
 *
 * The I2C devices will be created later, after the adapter for the relevant
 * bus has been registered.  After that moment, standard driver model tools
 * are used to bind "new style" I2C drivers to the devices.  The bus number
 * for any device declared using this routine is not available for dynamic
 * allocation.
 *
 * The board info passed can safely be __initdata, but be careful of embedded
 * pointers (for platform_data, functions, etc) since that won't be copied.
 */
int __init
i2c_register_board_info(int busnum,
    struct i2c_board_info const *info, unsigned len)
{
    int status;

    down_write(&__i2c_board_lock);

    /* dynamic bus numbers will be assigned after the last static one */
    if (busnum >= __i2c_first_dynamic_bus_num)
        __i2c_first_dynamic_bus_num = busnum + 1;

    for (status = 0; len; len--, info++) {
        struct i2c_devinfo  *devinfo;

        devinfo = kzalloc(sizeof(*devinfo), GFP_KERNEL);
        if (!devinfo) {
            pr_debug("i2c-core: can't register boardinfo!\n");
            status = -ENOMEM;
            break;
        }

        devinfo->busnum = busnum;
        devinfo->board_info = *info;
        list_add_tail(&devinfo->list, &__i2c_board_list);
    }

    up_write(&__i2c_board_lock);
```

```
51.
52.     return status;
53. }
```

i2c_register_board_info()把填写在struct i2c_board_info中的i2c slave device信息生成node，挂
到 `__i2c_board_list` list上。

in drivers/i2c/i2c-core.c

```
1.  static void i2c_scan_static_board_info(struct i2c_adapter *adapter)
2.  {
3.      struct i2c_devinfo  *devinfo;
4.
5.      down_read(&__i2c_board_lock);
6.      list_for_each_entry(devinfo, &__i2c_board_list, list) {
7.          if (devinfo->busnum == adapter->nr
8.                  && !i2c_new_device(adapter,
9.                          &devinfo->board_info))
10.             dev_err(&adapter->dev,
11.                 "Can't create device at 0x%02x\n",
12.                 devinfo->board_info.addr);
13.     }
14.     up_read(&__i2c_board_lock);
15. }
```

i2c_scan_static_board_info()扫描 `__i2c_board_list` list，把上面的node生成i2c device。

而 `i2c_scan_static_board_info()` 则是在 `i2c_register_adapter()` 中被调用。即每当有新的i2c bus注册，
则会扫描 `__i2c_board_list` list,看一下上面是否由该bus管理的device,是则create i2c device。

所以kernel支持M24C64和STTS2002 EEPROM的关键是create两个struct i2c_board_info，填写上对应的hardware
parameters，然后调用 `i2c_register_board_info()` 。

```
1.  /**
2.   * I2C_BOARD_INFO - macro used to list an i2c device and its address
3.   * @dev_type: identifies the device type
4.   * @dev_addr: the device's address on the bus.
5.   *
6.   * This macro initializes essential fields of a struct i2c_board_info,
7.   * declaring what has been provided on a particular board.  Optional
8.   * fields (such as associated irq, or device-specific platform_data)
9.   * are provided using conventional syntax.
10.  */
11. #define I2C_BOARD_INFO(dev_type, dev_addr) \
12.     .type = dev_type, .addr = (dev_addr)
```

I2C_BOARD_INFO macro用于帮组create struct i2c_board_info。

比如

```
struct i2c_board_info i2c_eeprom[2] = {
        I2C_BOARD_INFO("24c64", 0x50),
        I2C_BOARD_INFO("spd", 0x53),
};

i2c_register_board_info(0, i2c_eeprom, 2);
```

在gemstone2 toc board上，M24C64和STTS2002都挂在第一个i2c bus上。在原理图上是I2C1，但在Programmer
Guide上是I2C0。

Note:

必须enable CONFIG_I2C_BOARDINFO。

How to support i2c eeprom in kernel?

add following code into arch/arm/mach-pegmatite/pegmatite.c

```
static int __init i2c_eeprom_init(void)
{
    struct i2c_board_info i2c_eeprom[] =
    {
        {
            I2C_BOARD_INFO("24c64", 0x50),
            I2C_BOARD_INFO("spd", 0x53),
        },
    };

    i2c_register_board_info(0, i2c_eeprom, 2);

    return  0;
}
arch_initcall(i2c_eeprom_init);
```

make i2c_eeprom_init() before i2c-pxa driver init and after init-core driver init.

```
root@granite2:/sys/class/i2c-adapter/i2c-0# pwd
/sys/class/i2c-adapter/i2c-0
root@granite2:/sys/class/i2c-adapter/i2c-0# ls -l
drwxr-xr-x    3 root     root             0 May 23 06:41 0-0050
drwxr-xr-x    3 root     root             0 May 23 06:41 0-0053
--w-------    1 root     root          4096 May 23 06:43 delete_device
lrwxrwxrwx    1 root     root             0 May 23 06:43 device -> ../../d40
11000.i2c
-r--r--r--    1 root     root          4096 May 23 06:43 name
--w-------    1 root     root          4096 May 23 06:43 new_device
drwxr-xr-x    2 root     root             0 May 23 06:43 power
lrwxrwxrwx    1 root     root             0 May 23 06:41 subsystem -> ../../
../bus/i2c
-rw-r--r--    1 root     root          4096 May 23 06:41 uevent
```

这里的0-0050 and 0-0053就是两个eeprom device

```
root@granite2:/sys/class/i2c-adapter/i2c-0/0-0050# ls -l
lrwxrwxrwx    1 root     root             0 May 23 06:41 driver -> ../../../
../bus/i2c/drivers/at24
-rw-------    1 root     root          8192 May 23 06:51 eeprom
-r--r--r--    1 root     root          4096 May 23 06:43 modalias
-r--r--r--    1 root     root          4096 May 23 06:43 name
drwxr-xr-x    2 root     root             0 May 23 06:43 power
lrwxrwxrwx    1 root     root             0 May 23 06:41 subsystem -> ../../
../../bus/i2c
-rw-r--r--    1 root     root          4096 May 23 06:41 uevent
```

可以对eeprom进行read / write