

EEPROM driver porting

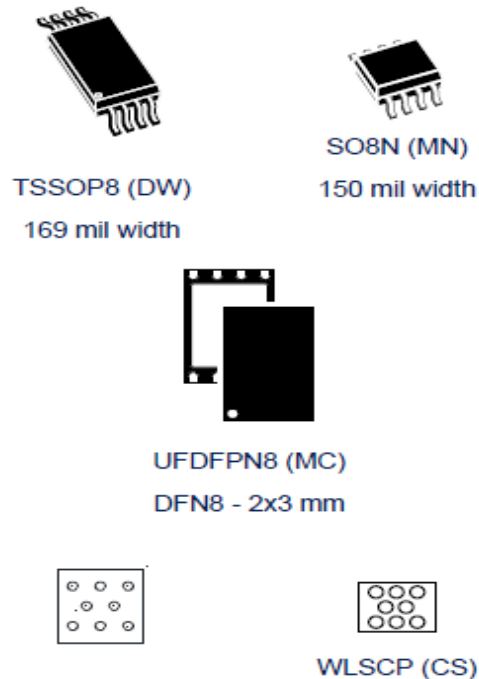
- I2C_tool verification
- Porting EEPROM driver
- Driver && eeprom
- User Space test case



M24512-W M24512-R M24512-DF

Datasheet

512-Kbit serial I²C bus EEPROM



Features

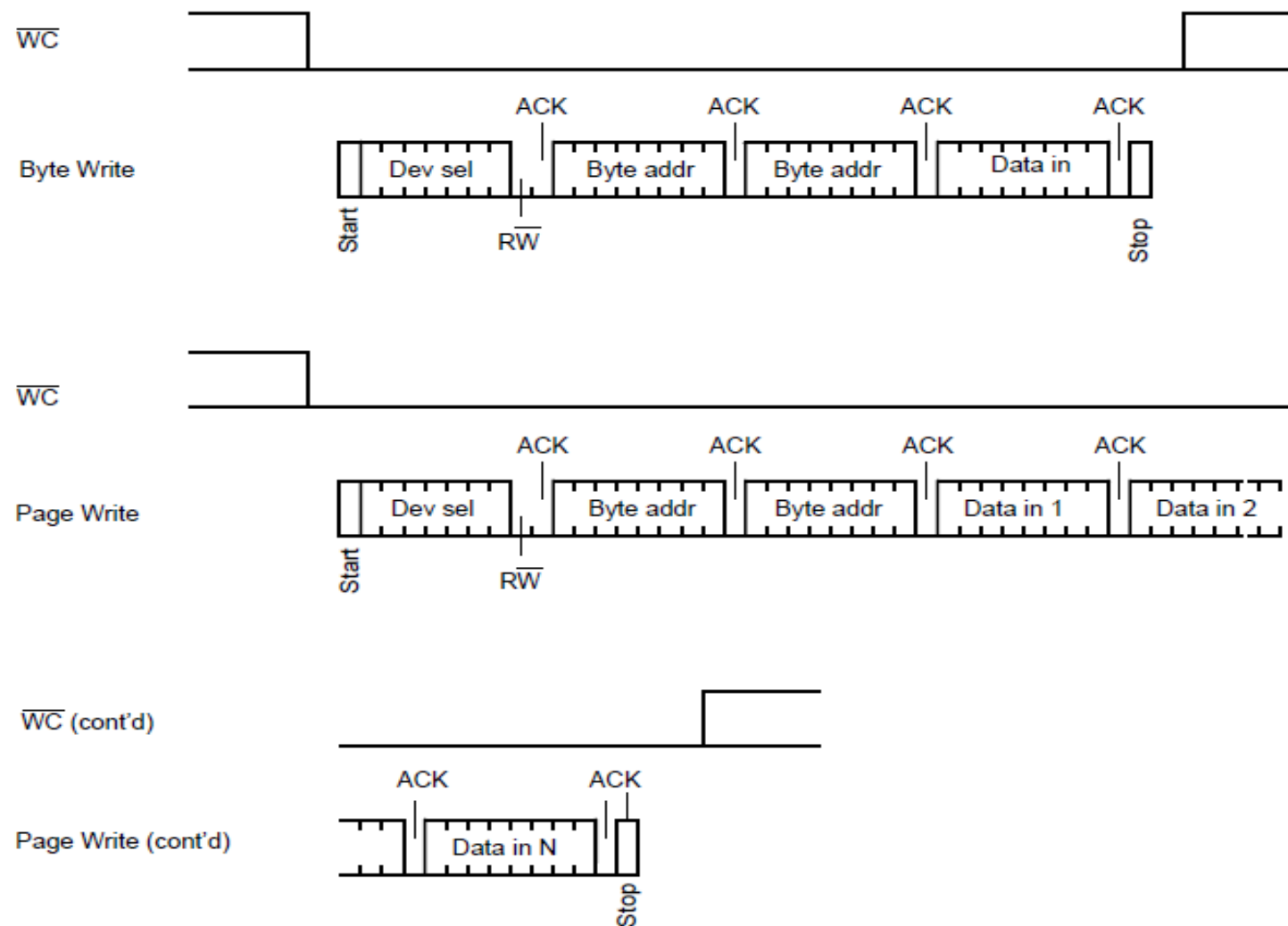
- Compatible with following I²C bus modes:
 - 1 MHz
 - 400 kHz
 - 100 kHz
- Memory array:
 - 512 Kbit (64 Kbyte) of EEPROM
 - Page size: 128 byte
 - Additional write lockable page (M24512-D order codes)
- Single supply voltage and high speed:
 - 1 MHz clock from 1.7 V to 5.5 V
- Write time:
 - Byte write within 5 ms
 - Page write within 5 ms
- Operating temperature range:
 - -40 °C up to +85 °C

5.1.1

Byte write

After the device select code and the address bytes, the bus master sends one data byte. If the addressed location is write-protected, by write control (\overline{WC}) being driven high, the device replies with NoAck, and the location is not modified. If, instead, the addressed location is not write-protected, the device replies with Ack. The bus master terminates the transfer by generating a stop condition, as shown in Figure 7.

Figure 7. Write mode sequences with $\overline{WC} = 0$ (data write enabled)



```
i2cdump -y 0 0x50
i2cset -f -y 0 0x50 0xe0 0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x11 0x12 0x13 0x14 0x15 0x16
i2cset -f -y 0 0x50 0xe0 0x00
i2cget -y 0 0x50

i2cset -f -y 0 0x50 0x00 0x10 0x10 0x11 0x12 0x13 0x14 0x15 0x16 0x17 0x18 0x19 0x1A 0x1B 0x1C 0x1D 0x1E 0x1F i
i2cset -f -y 0 0x50 0x00 0x10
i2cget -y 0 0x50
```

I2c bus = 0x50 位置

寫的時候 i2cset -f -y 0 0x50 0xe0 0x00

I2c 0 Bus 位置 Reg 位置

讀的時候 i2cset -f -y 0 0x50 0xe0 0x00

讀的時候要先把 i2ctool 的指針指向 0xe0 0x00

開始讀 i2cget -y 0 0x50 // 這時候指針就是指向0xe0 那個位置了

Porting EEPROM driver

[←](#) [→](#) [↺](#) [elixer.bootlin.com/linux/v4.19.112/source/Documentation/devicetree/bindings/eeprom/at24.txt](#)

[電影或是電視劇_文...](#) [茂金金典](#) [聯誼婚姻](#) [投資訊息](#) [生活日常](#) [工作相關事項](#) [鴻華先進_電動車](#) [好吃餐廳](#) [藝文活動導覽](#) [電腦程式_PC維修_...](#) [購買的電子書](#) [防疫相關訊息等等](#)

HOME ENGINEERING TRAINING DOCS COMMUNITY COMPANY

Real-Time Linux with PREEMPT_RT
Check our new training course with Creative Commons CC-BY-SA lecture and lab materials

bootlin
Elixir Cross Referencer

Linux

Filter tags

FIXME

v5

v4

v4.20

v4.19

v4.19.251

v4.19.250

v4.19.249

v4.19.248

v4.19.247

v4.19.246

v4.19.245

v4.19.244

v4.19.243

v4.19.242

v4.19.241

v4.19.240

v4.19.239

v4.19.238

v4.19.237

v4.19.236

v4.19.235

v4.19.234

v4.19.233

v4.19.232

v4.19.231

v4.19.230

v4.19.229

Documentation / devicetree / bindings / eeprom / at24.txt

All symbols

1 EEPROMs (I2C)

2

3 Required properties:

4

5 - compatible: Must be a "<manufacturer>,<model>" pair. The following <model>

6 values are supported (assuming "atmel" as manufacturer):

7

8 "atmel,24c00",

9 "atmel,24c01",

10 "atmel,24cs01",

11 "atmel,24c02",

12 "atmel,24cs02",

13 "atmel,24mac402",

14 "atmel,24mac602",

15 "atmel,spd",

16 "atmel,24c04",

17 "atmel,24cs04",

18 "atmel,24c08",

19 "atmel,24cs08",

20 "atmel,24c16",

21 "atmel,24cs16",

22 "atmel,24c32",

23 "atmel,24cs32",

24 "atmel,24c64",

25 "atmel,24cs64",

26 "atmel,24c128",

27 "atmel,24c256",

28 "atmel,24c512",

29 "atmel,24c1024",

30 "atmel,24c2048",

31

32 If <manufacturer> is not "atmel", then a fallback must be used

33 with the same <model> and "atmel" as manufacturer.

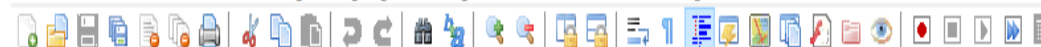
34

linux v4.19.112

```
43         "ramtron",
44         "renesas",
45         "rohm",
46         "st",
47
48         Some vendors use different model names for chips which are just
49         variants of the above. Known such exceptions are listed below:
50
51         "nxp,se97b" - the fallback is "atmel,24c02",
52         "renesas,r1ex24002" - the fallback is "atmel,24c02"
53         "renesas,r1ex24128" - the fallback is "atmel,24c128"
54         "rohm,br24t01" - the fallback is "atmel,24c01"
55
56     - reg: The I2C address of the EEPROM.
57
58 Optional properties:
59
60     - pagesize: The length of the pagesize for writing. Please consult the
61                 manual of your device, that value varies a lot. A wrong value
62                 may result in data loss! If not specified, a safety value of
63                 '1' is used which will be very slow.
64
65     - read-only: This parameterless property disables writes to the eeprom.
66
67     - size: Total eeprom size in bytes.
68
69     - no-read-rollover: This parameterless property indicates that the
70                         multi-address eeprom does not automatically roll over
71                         reads to the next slave address. Please consult the
72                         manual of your device.
73
74     - wp-gpios: GPIO to which the write-protect pin of the chip is connected.
75
76     - address-width: number of address bits (one of 8, 16).
77
78 Example:
79
80 eeprom@52 {
81     compatible = "atmel,24c32";
82     reg = <0x52>;
83     pagesize = <32>;
84     wp-gpios = <&gpio1 3 0>;
85 };
```

D:\技術文件\2022_05_16 以後所交付的任務和文件\驗政測試\RCON_EEPROM\nx-fe1114a4169733eb9a4b0596b6468f813ee4e460\defco

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?



i2c_xc24xx.c Kconfig Make.defs README.txt eeprom_testsample.c Daily review.txt new 2 defconfig

```
1328 # CONFIG_SENSORS_BH1770 is not set
1329 # CONFIG_SENSORS_APDS990X is not set
1330 # CONFIG_HMC6352 is not set
1331 # CONFIG_DS1682 is not set
1332 # CONFIG_LATTICE_ECP3_CONFIG is not set
1333 # CONFIG_SRAM is not set
1334 # CONFIG_PCI_ENDPOINT_TEST is not set
1335 # CONFIG_XILINX_SDFEC is not set
1336 # CONFIG_PVPANIC is not set
1337 # CONFIG_HISI_HIKEY_USB is not set
1338 # CONFIG_C2PORT is not set
1339
1340 #
1341 # EEPROM support
1342 #
1343 # CONFIG_EEPROM_AT24 is not set
1344 # CONFIG_EEPROM_AT25 is not set
1345 # CONFIG_EEPROM_LEGACY is not set
1346 # CONFIG_EEPROM_MAX6875 is not set
1347 # CONFIG_EEPROM_93CX6 is not set
1348 # CONFIG_EEPROM_93XX46 is not set
1349 # CONFIG_EEPROM_IDT_89HPESX is not set
1350 # CONFIG_EEPROM_EE1004 is not set
1351 # end of EEPROM support
1352
```



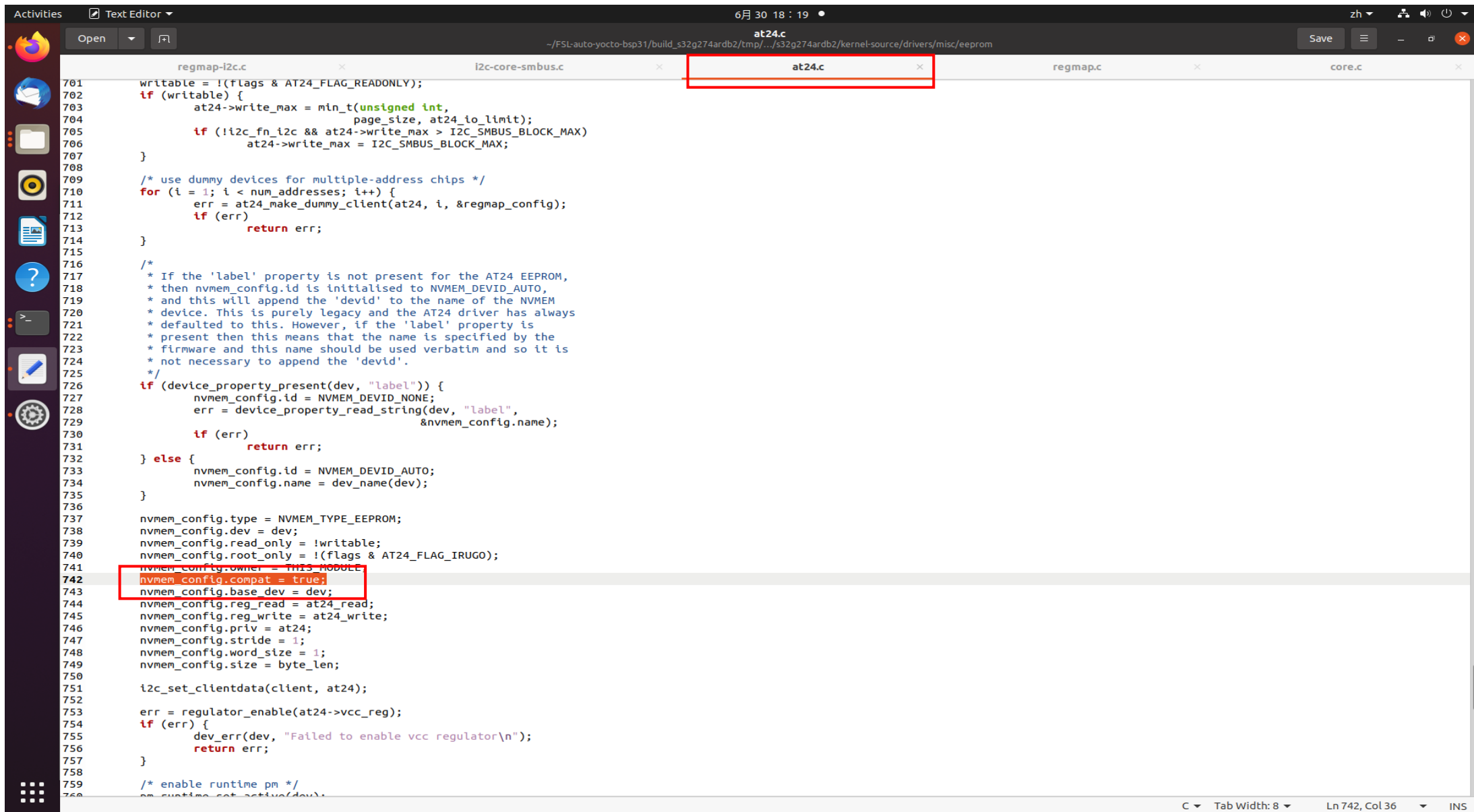
```
4 config EEPROM_AT24
5     tristate "I2C EEPROMs / RAMs / ROMs from most vendors"
6     depends on I2C && SYSFS
7     select NVMEM
8     select NVMEM_SYSFS
9     select REGMAP_I2C
10    help
11        Enable this driver to get read/write support to most I2C EEPROMs
12        and compatible devices like FRAMs, SRAMs, ROMs etc. After you
13        configure the driver to know about each chip on your target
14        board. Use these generic chip names, instead of vendor-specific
15        ones like at24c64, 24lc02 or fm24c04:
16
17        24c00, 24c01, 24c02, spd (readonly 24c02), 24c04, 24c08,
18        24c16, 24c32, 24c64, 24c128, 24c256, 24c512, 24c1024, 24c2048
19
20        Unless you like data loss puzzles, always be sure that any chip
21        you configure as a 24c32 (32 kbit) or larger is NOT really a
22        24c16 (16 kbit) or smaller, and vice versa. Marking the chip
23        as read-only won't help recover from this. Also, if your chip
24        has any software write-protect mechanism you may want to review the
25        code to make sure this driver won't turn it on by accident.
26
27        If you use this with an SMBus adapter instead of an I2C adapter,
28        full functionality is not available. Only smaller devices are
29        supported (24c16 and below, max 4 kByte).
30
31        This driver can also be built as a module. If so, the module
32        will be called at24.
33
```



```
cat /sys/bus/i2c/device/i2c-0/0-0050/name  
eeprom
```

It shall have Device node

Driver & eeprom



Activities Text Editor 6月 30 18:19 zh

Open at24.c

~/FSL-auto-yocto-bsp31/build_s32g274ardb2/tmp/.../s32g274ardb2/kernel-source/drivers/misc/eeprom

Save

regmap-i2c.c I2c-core-smbus.c at24.c regmap.c core.c

```
701 writable = !(flags & AT24_FLAG_READONLY);
702 if (writable) {
703     at24->write_max = min_t(unsigned int,
704                             page_size, at24->io_limit);
705     if (!i2c_fn_i2c && at24->write_max > I2C_SMBUS_BLOCK_MAX)
706         at24->write_max = I2C_SMBUS_BLOCK_MAX;
707 }
708
709 /* use dummy devices for multiple-address chips */
710 for (i = 1; i < num_addresses; i++) {
711     err = at24_make_dummy_client(at24, i, &regmap_config);
712     if (err)
713         return err;
714 }
715
716 /*
717  * If the 'label' property is not present for the AT24 EEPROM,
718  * then nvmem_config.id is initialised to NVMEM_DEVID_AUTO,
719  * and this will append the 'devid' to the name of the NVMEM
720  * device. This is purely legacy and the AT24 driver has always
721  * defaulted to this. However, if the 'label' property is
722  * present then this means that the name is specified by the
723  * firmware and this name should be used verbatim and so it is
724  * not necessary to append the 'devid'.
725  */
726 if (device_property_present(dev, "label")) {
727     nvmem_config.id = NVMEM_DEVID_NONE;
728     err = device_property_read_string(dev, "label",
729                                     &nvmem_config.name);
730     if (err)
731         return err;
732 } else {
733     nvmem_config.id = NVMEM_DEVID_AUTO;
734     nvmem_config.name = dev_name(dev);
735 }
736
737 nvmem_config.type = NVMEM_TYPE_EEPROM;
738 nvmem_config.dev = dev;
739 nvmem_config.read_only = !writable;
740 nvmem_config.root_only = !(flags & AT24_FLAG_IRUGO);
741 nvmem_config.owner = THIS_MODULE;
742 nvmem_config.compat = true;
743 nvmem_config.base_dev = dev;
744 nvmem_config.reg_read = at24_read;
745 nvmem_config.reg_write = at24_write;
746 nvmem_config.priv = at24;
747 nvmem_config.stride = 1;
748 nvmem_config.word_size = 1;
749 nvmem_config.size = byte_len;
750
751 i2c_set_clientdata(client, at24);
752
753 err = regulator_enable(at24->vcc_reg);
754 if (err) {
755     dev_err(dev, "Failed to enable vcc regulator\n");
756     return err;
757 }
758
759 /* enable runtime pm */
760 pm_runtime_set_active(dev);
```

C Tab Width: 8 Ln 742, Col 36 INS



Open

at24.c
~/FSL-auto-yocto-bsp31/build_s32g274ardb2/tmp/.../s32g274ardb2/kernel-source/drivers/misc/eeeprom

Save



regmap-i2c.c

i2c-core-smbus.c

at24.c

regmap.c

core.c

```
736 nvmem_config.type = NVMEM_TYPE_EEPROM;
737 nvmem_config.dev = dev;
738 nvmem_config.read_only = !writable;
740 nvmem_config.root_only = !(flags & AT24_FLAG_IRUGO);
741 nvmem_config.owner = THIS_MODULE;
742 nvmem_config.compat = true;
743 nvmem_config.base_dev = dev;
744 nvmem_config.reg_read = at24_read;
745 nvmem_config.reg_write = at24_write;
746 nvmem_config.priv = at24;
747 nvmem_config.stride = 1;
748 nvmem_config.word_size = 1;
749 nvmem_config.size = byte_len;
750
751 i2c_set_clientdata(client, at24);
752
753 err = regulator_enable(at24->vcc_reg);
754 if (err) {
755     dev_err(dev, "Failed to enable vcc regulator\n");
756     return err;
757 }
758
759 /* enable runtime pm */
760 pm_runtime_set_active(dev);
761 pm_runtime_enable(dev);
762
763 at24->nvmem = devnvmem_register(dev, &nvmem_config);
764 if (IS_ERR(at24->nvmem)) {
765     pm_runtime_disable(dev);
766     if (!pm_runtime_status_suspended(dev))
767         regulator_disable(at24->vcc_reg);
768     return PTR_ERR(at24->nvmem);
769 }
770
771 /*
772  * Perform a one-byte test read to verify that the
773  * chip is functional.
774  */
775 err = at24_read(at24, 0, &test_byte, 1);
776 if (err) {
777     pm_runtime_disable(dev);
778     if (!pm_runtime_status_suspended(dev))
779         regulator_disable(at24->vcc_reg);
780     return -ENODEV;
781 }
782
783 pm_runtime_idle(dev);
784
785 if (writable)
786     dev_info(dev, "%u byte %s EEPROM, writable, %u bytes/write\n",
787              byte_len, client->name, at24->write_max);
788 else
789     dev_info(dev, "%u byte %s EEPROM, read-only\n",
790              byte_len, client->name);
791
792 return 0;
793 }
794
795 static int at24_remove(struct i2c_client *client)
```



Open



~/FSL-auto-yocto-bsp31/build_s32g274ardb2/tmp/...shared/s32g274ardb2/kernel-source/drivers/nvmem

Save



regmap-i2c.c

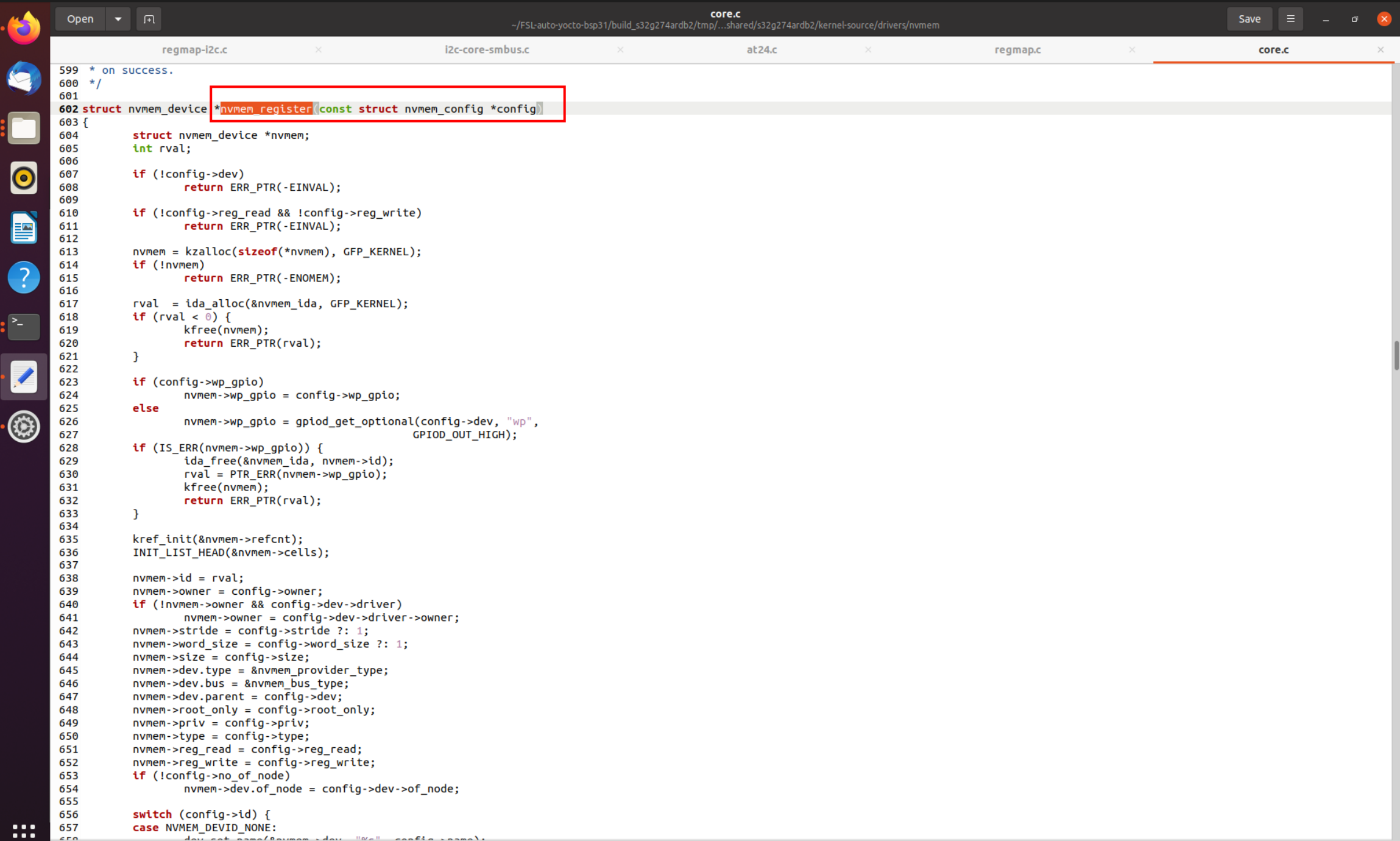
i2c-core-smbus.c

at24.c

regmap.c

core.c

```
731
732     nvmem_device_remove_all_cells(nvmem);
733     device_unregister(&nvmem->dev);
734 }
735
736 /**
737  * nvmem_unregister() - Unregister previously registered nvmem device
738  *
739  * @nvmem: Pointer to previously registered nvmem device.
740  */
741 void nvmem_unregister(struct nvmem_device *nvmem)
742 {
743     kref_put(&nvmem->refcnt, nvmem_device_release);
744 }
745 EXPORT_SYMBOL_GPL(nvmem_unregister);
746
747 static void devm_nvmem_release(struct device *dev, void *res)
748 {
749     nvmem_unregister(*(struct nvmem_device **)res);
750 }
751
752 /**
753  * devm_nvmem_register() - Register a managed nvmem device for given
754  * nvmem_config.
755  * Also creates a binary entry in /sys/bus/nvmem/devices/dev-name/nvmem
756  *
757  * @dev: Device that uses the nvmem device.
758  * @config: nvmem device configuration with which nvmem device is created.
759  *
760  * Return: Will be an ERR_PTR() on error or a valid pointer to nvmem_device
761  * on success.
762  */
763 struct nvmem_device *devm_nvmem_register(struct device *dev,
764                                          const struct nvmem_config *config)
765 {
766     struct nvmem_device **ptr, *nvmem;
767
768     ptr = devres_alloc(devm_nvmem_release, sizeof(*ptr), GFP_KERNEL);
769     if (!ptr)
770         return ERR_PTR(-ENOMEM);
771
772     nvmem = nvmem_register(config);
773
774     if (!IS_ERR(nvmem)) {
775         *ptr = nvmem;
776         devres_add(dev, ptr);
777     } else {
778         devres_free(ptr);
779     }
780
781     return nvmem;
782 }
783 EXPORT_SYMBOL_GPL(devm_nvmem_register);
784
785 static int devm_nvmem_match(struct device *dev, void *res, void *data)
786 {
787     struct nvmem_device **r = res;
788
789     return *r == data;
```



The image shows a Linux desktop environment with a text editor window open. The window title is "core.c" and the path is "~/FSL-auto-yocto-bsp31/build_s32g274ardb2/tmp/...shared/s32g274ardb2/kernel-source/drivers/nvmem". The editor displays the source code for "core.c". A red box highlights the line: `602 struct nvmem_device *nvmem_register(const struct nvmem_config *config)`. The code includes various kernel API calls and error handling. The status bar at the bottom indicates "Bracket match found on line: 602".

```
599 * on success.
600 */
601
602 struct nvmem_device *nvmem_register(const struct nvmem_config *config)
603 {
604     struct nvmem_device *nvmem;
605     int rval;
606
607     if (!config->dev)
608         return ERR_PTR(-EINVAL);
609
610     if (!config->reg_read && !config->reg_write)
611         return ERR_PTR(-EINVAL);
612
613     nvmem = kzalloc(sizeof(*nvmem), GFP_KERNEL);
614     if (!nvmem)
615         return ERR_PTR(-ENOMEM);
616
617     rval = ida_alloc(&nvmem_ida, GFP_KERNEL);
618     if (rval < 0) {
619         kfree(nvmem);
620         return ERR_PTR(rval);
621     }
622
623     if (config->wp_gpio)
624         nvmem->wp_gpio = config->wp_gpio;
625     else
626         nvmem->wp_gpio = gpiod_get_optional(config->dev, "wp",
627                                             GPIOD_OUT_HIGH);
628     if (IS_ERR(nvmem->wp_gpio)) {
629         ida_free(&nvmem_ida, nvmem->id);
630         rval = PTR_ERR(nvmem->wp_gpio);
631         kfree(nvmem);
632         return ERR_PTR(rval);
633     }
634
635     kref_init(&nvmem->refcnt);
636     INIT_LIST_HEAD(&nvmem->cells);
637
638     nvmem->id = rval;
639     nvmem->owner = config->owner;
640     if (!nvmem->owner && config->dev->driver)
641         nvmem->owner = config->dev->driver->owner;
642     nvmem->stride = config->stride ?: 1;
643     nvmem->word_size = config->word_size ?: 1;
644     nvmem->size = config->size;
645     nvmem->dev.type = &nvmem_provider_type;
646     nvmem->dev.bus = &nvmem_bus_type;
647     nvmem->dev.parent = config->dev;
648     nvmem->root_only = config->root_only;
649     nvmem->priv = config->priv;
650     nvmem->type = config->type;
651     nvmem->reg_read = config->reg_read;
652     nvmem->reg_write = config->reg_write;
653     if (!config->no_of_node)
654         nvmem->dev.of_node = config->dev->of_node;
655
656     switch (config->id) {
657     case NVMEM_DEVID_NONE:
658         dev_set_name(&nvmem->dev, "%s-%s-%s", config->name,
```

regmap-i2c.c

i2c-core-smbus.c

at24.c

regmap.c

core.c

```
657     case NVMEM_DEVID_NONE:
658         dev_set_name(&nvmem->dev, "%s", config->name);
659         break;
660     case NVMEM_DEVID_AUTO:
661         dev_set_name(&nvmem->dev, "%s%d", config->name, nvmem->id);
662         break;
663     default:
664         dev_set_name(&nvmem->dev, "%s%d",
665                     config->name ? : "nvmem",
666                     config->name ? config->id : nvmem->id);
667         break;
668 }
669
670 nvmem->read_only = device_property_present(config->dev, "read-only") ||
671                 config->read_only || !nvmem->reg_write;
672
673 #ifdef CONFIG_NVMEM_SYSFS
674 nvmem->dev.groups = nvmem_dev_groups;
675 #endif
676
677 dev_dbg(&nvmem->dev, "Registering nvmem device %s\n", config->name);
678
679 rval = device_register(&nvmem->dev);
680 if (rval)
681     goto err_put_device;
682
683 if (config->compat) {
684     rval = nvmem_sysfs_setup_compat(nvmem, config);
685     if (rval)
686         goto err_device_del;
687 }
688
689 if (config->cells) {
690     rval = nvmem_add_cells(nvmem, config->cells, config->ncells);
691     if (rval)
692         goto err_teardown_compat;
693 }
694
695 rval = nvmem_add_cells_from_table(nvmem);
696 if (rval)
697     goto err_remove_cells;
698
699 rval = nvmem_add_cells_from_of(nvmem);
700 if (rval)
701     goto err_remove_cells;
702
703 blocking_notifier_call_chain(&nvmem_notifier, NVMEM_ADD, nvmem);
704
705 return nvmem;
706
707 err_remove_cells:
708     nvmem_device_remove_all_cells(nvmem);
709 err_teardown_compat:
710     if (config->compat)
711         nvmem_sysfs_remove_compat(nvmem, config);
712 err_device_del:
713     device_del(&nvmem->dev);
714 err_put_device:
715     put_device(&nvmem->dev);
716
```

Bracket match found on line: 684

C Tab Width: 8

Ln 684, Col 48

INS

```
core.c
~/FSL-auto-yocto-bsp31/build_s32g274ardb2/tmp/...shared/s32g274ardb2/kernel-source/drivers/nvmem

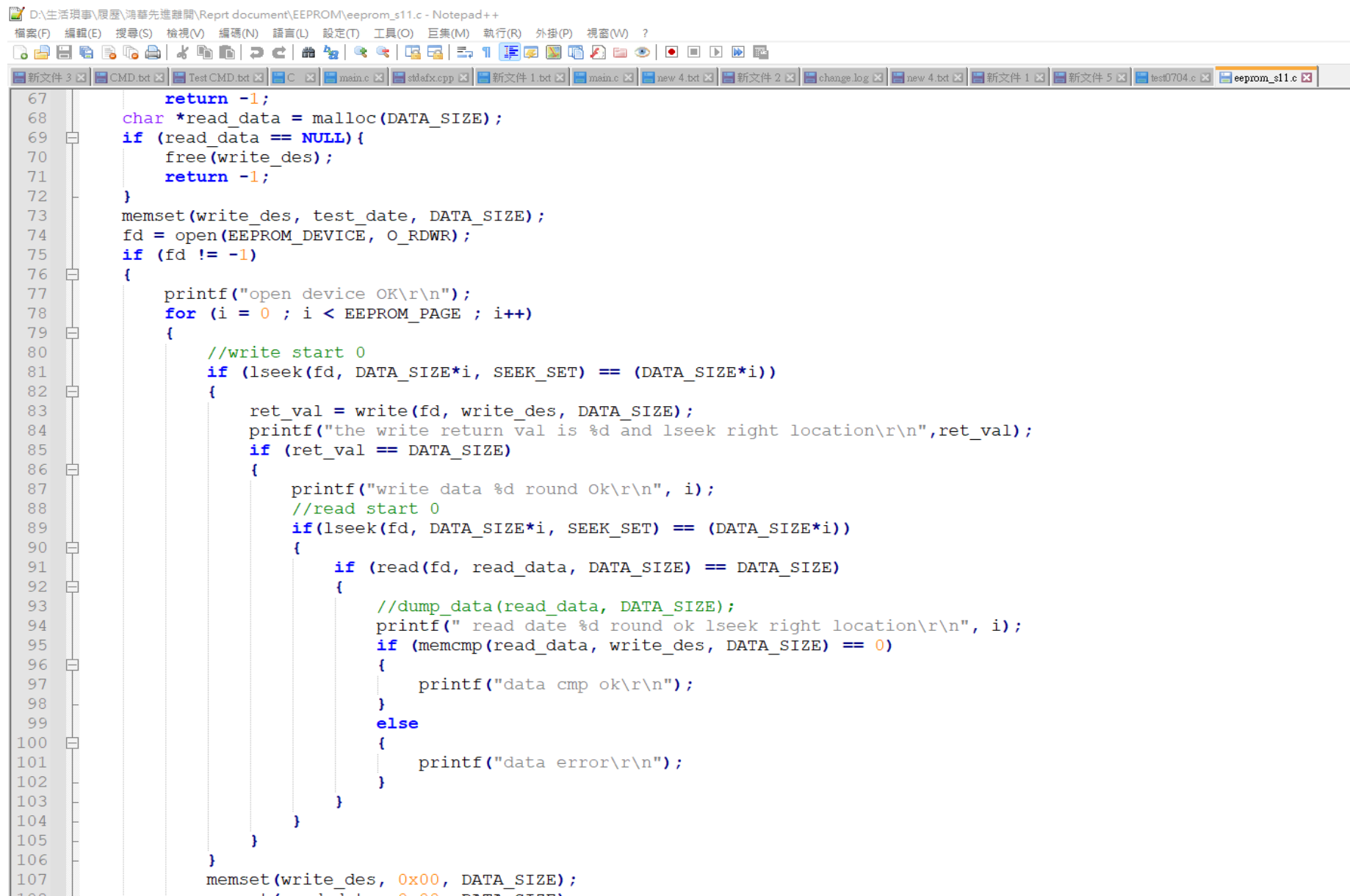
regmap-i2c.c  i2c-core-smbus.c  at24.c  regmap.c  core.c

252 };
253
254 static struct bin_attribute bin_attr_nvmem_eeprom_compat = {
255     .attr = {
256         .name = "eeprom",
257     },
258     .read = bin_attr_nvmem_read,
259     .write = bin_attr_nvmem_write,
260 };
261
262 /*
263  * nvmem_setup_compat() - Create an additional binary entry in
264  * drivers sys directory, to be backwards compatible with the older
265  * drivers/misc/eeprom drivers.
266  */
267 static int nvmem_sysfs_setup_compat(struct nvmem_device *nvmem,
268                                     const struct nvmem_config *config)
269 {
270     int rval;
271
272     if (!config->compat)
273         return 0;
274
275     if (!config->base_dev)
276         return -EINVAL;
277
278     nvmem->eeprom = bin_attr_nvmem_eeprom_compat;
279     nvmem->eeprom.attr.mode = nvmem_bin_attr_get_umode(nvmem);
280     nvmem->eeprom.size = nvmem->size;
281 #ifdef CONFIG_DEBUG_LOCK_ALLOC
282     nvmem->eeprom.attr.key = &eeprom_lock_key;
283 #endif
284     nvmem->eeprom.private = &nvmem->dev;
285     nvmem->base_dev = config->base_dev;
286
287     rval = device_create_bin_file(nvmem->base_dev, &nvmem->eeprom);
288     if (rval) {
289         dev_err(&nvmem->dev,
290               "Failed to create eeprom binary file %d\n", rval);
291         return rval;
292     }
293
294     nvmem->flags |= FLAG_COMPAT;
295
296     return 0;
297 }
298
299 static void nvmem_sysfs_remove_compat(struct nvmem_device *nvmem,
300                                       const struct nvmem_config *config)
301 {
302     if (config->compat)
303         device_remove_bin_file(nvmem->base_dev, &nvmem->eeprom);
304 }
305
306 #else /* CONFIG_NVMEM_SYSFS */
307
308 static int nvmem_sysfs_setup_compat(struct nvmem_device *nvmem,
309                                     const struct nvmem_config *config)
310 {
311     return -ENOSYS;

```

Here is .

User Space test case



```
D:\生活瑣事\履歷\鴻華先進離職\Reprt document\EEPROM\eeeprom_s11.c - Notepad++
檔案(F) 編輯(E) 搜尋(S) 檢視(V) 編碼(N) 語言(L) 設定(T) 工具(O) 巨集(M) 執行(R) 外掛(P) 視窗(W) ?
新文件 3  CMD.txt  TestCMD.txt  C  main.c  stdafx.cpp  新文件 1.txt  main.c  new 4.txt  新文件 2  change.log  new 4.txt  新文件 1  新文件 5  test0704.c  eeeprom_s11.c

67     return -1;
68     char *read_data = malloc(DATA_SIZE);
69     if (read_data == NULL){
70         free(write_des);
71         return -1;
72     }
73     memset(write_des, test_date, DATA_SIZE);
74     fd = open(EEPROM_DEVICE, O_RDWR);
75     if (fd != -1)
76     {
77         printf("open device OK\r\n");
78         for (i = 0 ; i < EEPROM_PAGE ; i++)
79         {
80             //write start 0
81             if (lseek(fd, DATA_SIZE*i, SEEK_SET) == (DATA_SIZE*i))
82             {
83                 ret_val = write(fd, write_des, DATA_SIZE);
84                 printf("the write return val is %d and lseek right location\r\n",ret_val);
85                 if (ret_val == DATA_SIZE)
86                 {
87                     printf("write data %d round Ok\r\n", i);
88                     //read start 0
89                     if(lseek(fd, DATA_SIZE*i, SEEK_SET) == (DATA_SIZE*i))
90                     {
91                         if (read(fd, read_data, DATA_SIZE) == DATA_SIZE)
92                         {
93                             //dump_data(read_data, DATA_SIZE);
94                             printf(" read date %d round ok lseek right location\r\n", i);
95                             if (memcmp(read_data, write_des, DATA_SIZE) == 0)
96                             {
97                                 printf("data cmp ok\r\n");
98                             }
99                             else
100                             {
101                                 printf("data error\r\n");
102                             }
103                         }
104                     }
105                 }
106             }
107         }
108         memset(write_des, 0x00, DATA_SIZE);
109         memset(read_data, 0x00, DATA_SIZE);
110     }
```