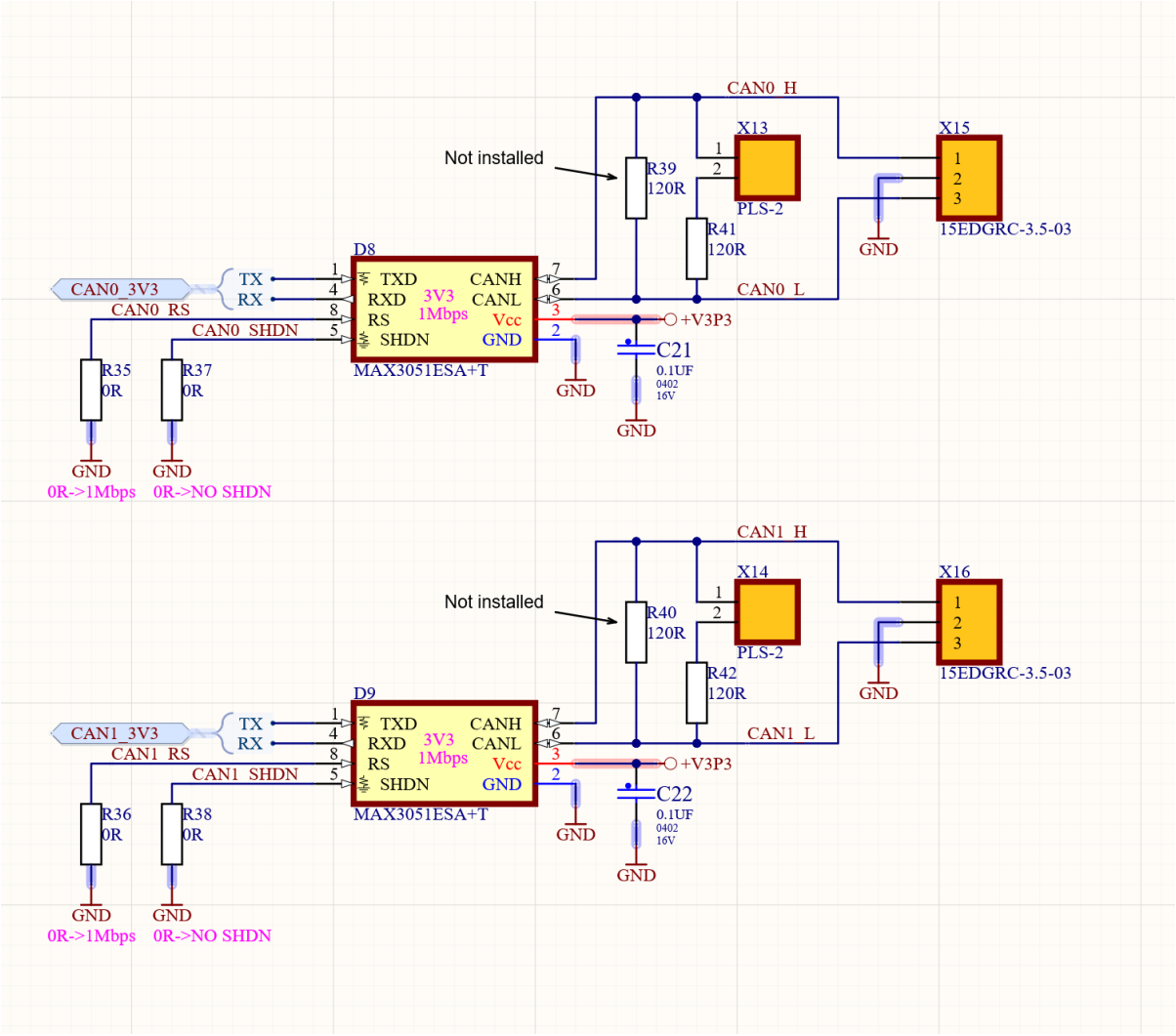
Содержание

RK3568 Quick Start Kernel 6.1

CAN

Для корректной работы шины CAN на отладочной плате NMS_SM_EVM необходимо подключить терминаторы - установить джамперы X13 и X14.



Драйвер CAN собирается как модуль ядра .ko с помощью buildroot-external. Исходники взяты из ядра 4.19, которое предоставляется Rockchip. Модуль ядра подгружается в **/etc/init.d/S49CANinit**.

Пример настройки интерфейса и проверки в режиме loopback CAN0↔CAN1:

```
#configure
ip link set dev can0 down
ip link set dev can1 down
ip link set can0 up type can bitrate 1000000 dbitrate 1000000 fd on
ip link set can1 up type can bitrate 1000000 dbitrate 1000000 fd on

#test
candump -T 1000 can0 > /tmp/1.txt&
sleep 1
cansend can1 500#DEADBEEF
```

```
sleep 2

cat /tmp/1.txt
```

CAN в ядре 6.1 работает только в режиме

fd on

```
# candump -T 1000 can0 > /tmp/1.txt&
#
# usleep 100000
# cansend can1 500#1E.10.10
# cat /tmp/1.txt
  can0  500   [3]  1E 10 10
#
```

ETH LED configuration

Светодиоды Ethernet управляются PHY DAP8211. Описания регистров п. 4.3.10 - 4.3.14

PHY управляется через MDIO регистры. Для доступа к регистрам используется утилита mdio-tool

Связь SMARC-пинов и регистров

SMARC Pin	SMARC Name	PHY LED	PHY Reg
P21	GBE0_LINK100#	LED0	0xA00C
P22	GBE0_LINK1000#	LED1	0xA00D
P25	GBE0_LINK_ACT#	LED2	0xA00E
S19	GBE1_LINK100#	LED0	0xA00C
S22	GBE1_LINK1000#	LED1	0xA00D
S31	GBE1_LINK_ACT#	LED2	0xA00E

Для работы с регистрами светодиодов используется General Extend Mapping:

Offset	Name	Description
0x1E	EXT_ADD	Extended Register Address Registe
0x1F	EXT_DATA	Extended Register Data Register

В EXT_ADD требуется записать адрес желаемого Extended Register, в/из EXT_DATA можно писать/читать данные.

Пример настройки на NMS-SM-EVM

На отладочной плате на ETH заведены GBE0_LINK1000 - на YELLOW LED, GBE1_LINK_ACT - на GREEN

По умолчанию PHY настроен на режим 0x0620 - Yellow ACT 1Gbit/s; Green ACT 100Mbit/s;

Пример настройки, при котором Yellow и Green будут ACT на скорости 1000:

```
./mdio-tool w eth0 0x1e 0xA00D #set EXT reg
./mdio-tool r eth0 0x1f #read EXT reg
./mdio-tool w eth0 0x1f 0x0660 #write new value
./mdio-tool w eth0 0x1e 0x0 #set EXT_ADDR reg to default
```

Работа с GPIO

Стандарт модулей SMARC предусматривает GPIO0-GPIO13, но не все они имплементированы в модулях NMS-SM-RK3568:

- В ревизии 1 реализованы GPIO0-GPIO3
- В ревизии 2 реализованы GPIO0-GPIO3, GPIO10, GPIO11
- В ревизии 3 реализованы все GPIO стандарта SMARC. (ссылка на dts)

Также при необходимости некоторые интерфейсные пины(в соответвии с таблицей «Распиновка разъема») можно переопределить как GPIO.

Процессор содержит 5 банков по 32 пина. Все GPIO прописаны dts и видны */sys/kernel/debug/gpio*. Для управления состоянием пина по имени (имя во 2 столбце) можно использовать скрипт

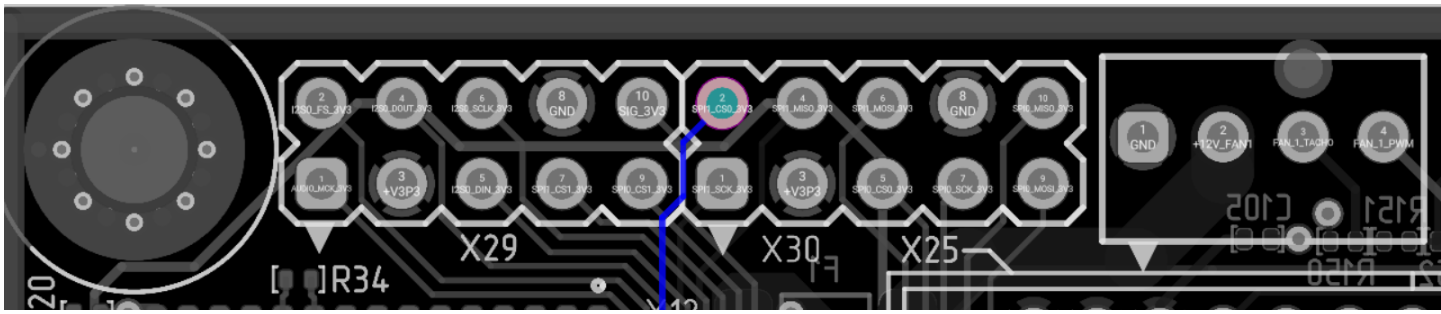
gpio.sh

```
root@192-168-1-101:~# cat /sys/kernel/debug/gpio
gpiochip0: GPIOs 0-31, parent: platform/fdd60000.gpio, gpio0:
 gpio-0   (GPIO0_A0            )
 gpio-1   (GPIO0_A1            )
 gpio-2   (GPIO0_A2            )
 .......
 gpio-30  (GPIO0_D6            )
 gpio-31  (GPIO0_D7            )

gpiochip1: GPIOs 32-63, parent: platform/fe740000.gpio, gpio1:
 gpio-32  (GPIO1_A0            )
 gpio-33  (GPIO1_A1            )
 gpio-34  (GPIO1_A2            )
 ........
 gpio-62  (GPIO1_D6            )
 gpio-63  (GPIO1_D7            )

gpiochip2: GPIOs 64-95, parent: platform/fe750000.gpio, gpio2:
 gpio-64  (GPIO2_A0            )
```

```
 gpio-65  (GPIO2_A1            )
 gpio-66  (GPIO2_A2            )
 .........
gpiochip3: GPIOs 96-127, parent: platform/fe760000.gpio, gpio3:
 gpio-96  (GPIO3_A0            )
 gpio-97  (GPIO3_A1            |reset               ) out hi
 gpio-98  (GPIO3_A2            )
 .....
 gpio-126 (GPIO3_D6            )
 gpio-127 (GPIO3_D7            )

gpiochip4: GPIOs 128-159, parent: platform/fe770000.gpio, gpio4:
 gpio-128 (GPIO4_A0            )
 gpio-129 (GPIO4_A1            )
 gpio-130 (GPIO4_A2            )
 ........
 gpio-158 (GPIO4_D6            )
 gpio-159 (GPIO4_D7            )
```

Пример использования интерфейсного пина в качестве GPIO

Например, переопределим SPI0_CS как GPIO:



Необходимо в таблице «Распиновка разъема» узнать, доступна ли функция GPIO в качестве альтернативной:

Распиновка разъемов

Распиновка разъема X1 согласно SMARC

Search

Pin #	Name STANDART	Signals_SM_RK3568	CPU PIN	CPU Function IO description Func1 Func2 Func3 Func4 Func5 Func6	Group	I/O Type	I/O Level	PU / PD SM_RK3568	Description
	SPI_CS0								
P54	ESPI_CS0# / SPI1_CS0# / QSPI_CS0#	SPI1.CS0	AE8	GPIO4_C6_d PWM13_M1 SPI3_CS0_M1 SATA0_ACT_LED UART9_RX_M1 I2S3_SDI_M1	SPI1	O CMOS	1V8		SPI1 Master Chip Select

Управление пином:

```
root@192-168-1-101:~# cat /sys/kernel/debug/gpio | grep 4_C6 #проверка
```

```
наличия
 gpio-150 (GPIO4_C6            )
root@192-168-1-101:~# ./gpio.sh GPIO4_C6 1 #запись в единицу
root@192-168-1-101:~# cat /sys/kernel/debug/gpio | grep 4_C6
 gpio-150 (GPIO4_C6            |sysfs               ) out hi
root@192-168-1-101:~# ./gpio.sh GPIO4_C6 0 #запипсь в ноль
root@192-168-1-101:~# cat /sys/kernel/debug/gpio | grep 4_C6
 gpio-150 (GPIO4_C6            |sysfs               ) out lo
```

I2C-GPIO расширитель на отладочной плате SM_EVM

Отладочная плата содержит PCA9535PW расширитель на адресе 0x20, подключенный к i2c-контроллеру с адресом fe5c0000. В этой dtb прописаны I2C-GPIO в соответвии со схемой. Исходники dtb и инструкция по сборке

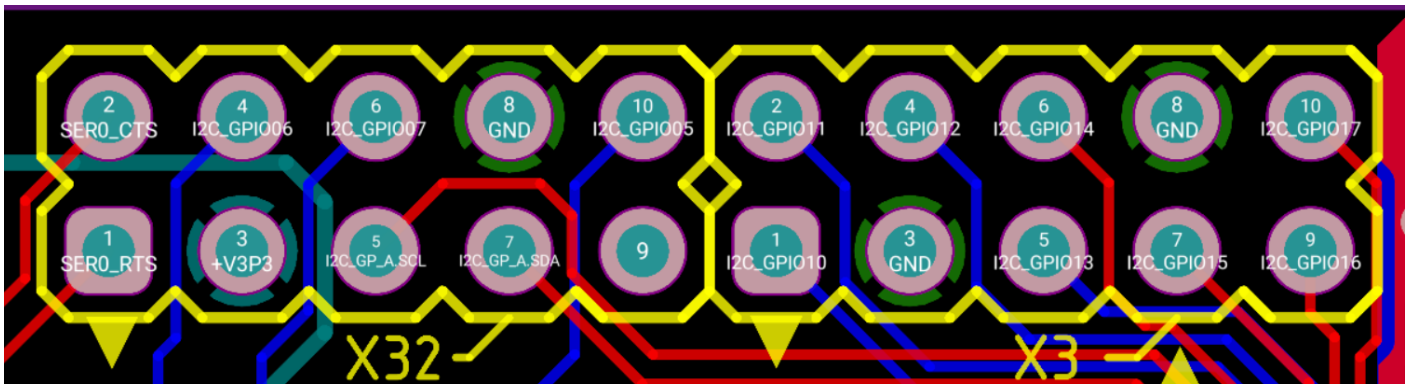
```
gpiochip5: GPIOs 496-511, parent: i2c/3-0020, 3-0020, can sleep:
 gpio-496 (DSI0_TP_RST_N       )
 gpio-497 (DSI1_TP_RST_N       )
 gpio-498 (DSI0_RST_N          )
 gpio-499 (DSI1_RST_N          )
 gpio-500 (CAM0_PWDN           )
 gpio-501 (I2C_GPIO05          |sysfs               ) out lo
 gpio-502 (I2C_GPIO06          |sysfs               ) out hi
 gpio-503 (I2C_GPIO07          )
 gpio-504 (I2C_GPIO10          )
 gpio-505 (I2C_GPIO11          )
 gpio-506 (I2C_GPIO12          )
 gpio-507 (I2C_GPIO13          )
 gpio-508 (I2C_GPIO14          )
 gpio-509 (I2C_GPIO15          )
 gpio-510 (I2C_GPIO16          )
 gpio-511 (I2C_GPIO17          )
```

Фрагмент dts с описание gpio-line-names:

```
        i2cgpio: pca9535@20 {
                compatible = "nxp,pca9535";
                /* vcc-supply = <&regulator_i2c_1v8 */
                reg = <0x20>;
                /*GPIO.3 -> P111 (som) -> gpio3_d2*/
                interrupt-parent = <&gpio3>;
                interrupts = <27 IRQ_TYPE_LEVEL_LOW>;
                interrupt-controller;
                #interrupt-cells = <2>;

                gpio-controller;
                #gpio-cells = <2>;
                gpio-line-names =
```

```
"DSI0_TP_RST_N","DSI1_TP_RST_N","DSI0_RST_N","DSI1_RST_N","CAM0_PWDN","
I2C_GPIO05","I2C_GPIO06","I2C_GPIO07",
"I2C_GPIO10","I2C_GPIO11","I2C_GPIO12","I2C_GPIO13","I2C_GPIO14","I2C_G
PIO15","I2C_GPIO16","I2C_GPIO17";
        };
```

Выводы GPIO на headers:



Пример управления I2C_GPIO5:

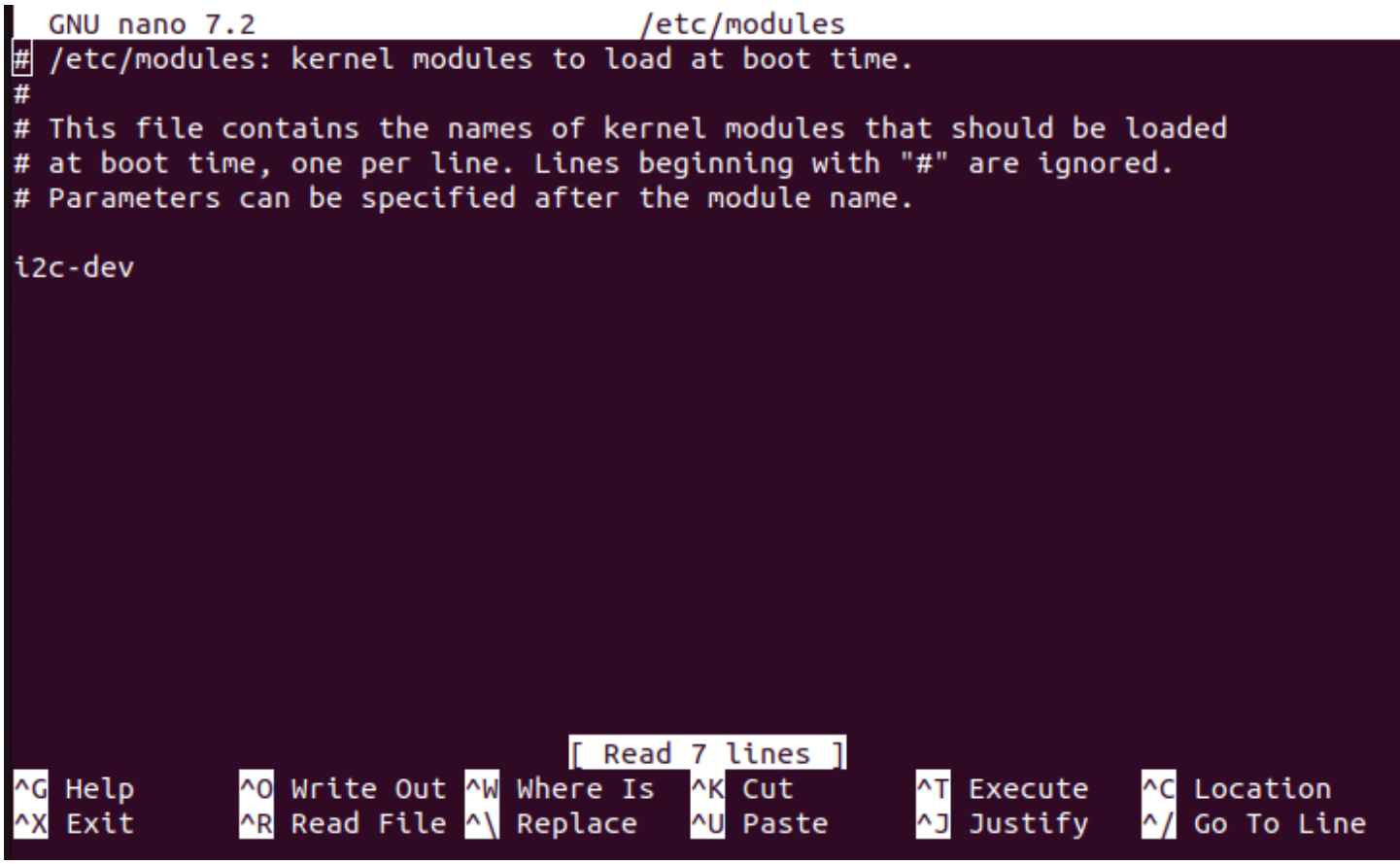
```
root@192-168-1-101:~# ./gpio.sh I2C_GPIO05 1 #записать единицу
root@192-168-1-101:~# ./gpio.sh I2C_GPIO05 0 #записать ноль
```

I2C в Debian

Если в Debian при попытке *i2cdetect* шин выходит ошибка, то нужно подгрузить Kernel Module *i2c-dev*.

```
root@192-168-1-101:~# i2cdetect -y 0
Error: Could not open file `/dev/i2c-0' or `/dev/i2c/0': No such file
or directory
```

Для этого необходимо в */etc/modules* добавить имя модуля и перезагрузить устройство:



После перезагрузки в dev появятся i2c-линии:

```
root@192-168-1-101:~# i2cdetect -y 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: UU -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: 60 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

SPI в Debian

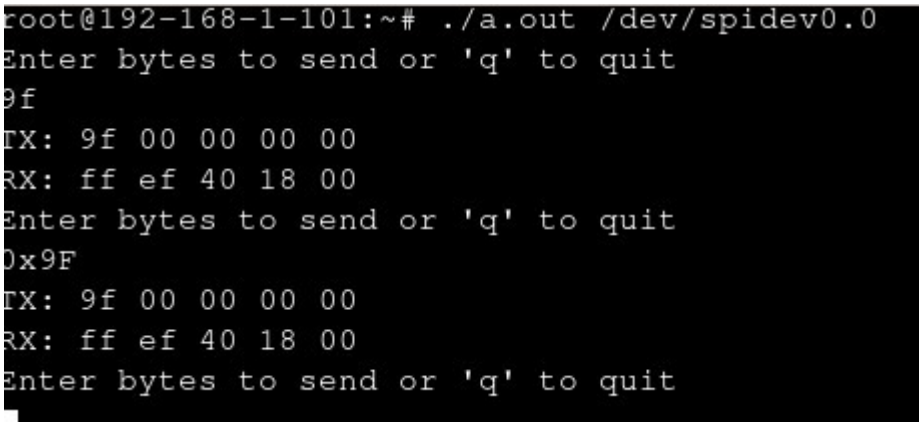
Для общения с SPI устройствами в ядре должен быть модуль spidev.

В качестве примера в */root/* лежит программа *run_spi* и ее исходный код *spi.c* (либо на github), которые реализуют коммуникацию с SPI устройством на примере SPI FLASH W25Q128FV.

Инструкция JEDEC ID для данного устройства имеет вид: 0x9F [MF7-MF0] [ID15-ID8] [ID7-ID0].

То есть, первый байт последовательности указывается нами - 0x9F. Далее получаем 3 оставшихся байта.

Скриншот выполнения инструкции JEDEC ID для SPI FLASH:



Как видно на скриншоте, вводить байты можно как с 0x, так и без.

Если использовать несколько байт подряд, то указывать их оптимально через пробел (другие разделители не использовать)