FOSDEM 2024

Linux CAN upstreaming on MMU-less systems

Amarula Solutions

Dario Binacchi
dario.binacchi@amarulasolutions.com
https://github.com/passgat

Dario Binacchi

- ➢ Embedded Linux engineer at **Amarula Solutions**:
  - Embedded Linux and Android expertise
  - Development, consulting and training
  - Open source projects
- ➢ Open source contributor
  - Buildroot
  - Linux
    - Contributor to slCAN driver
    - Developed the bxCAN driver
  - U-Boot
    - Custodian for NAND subsystem
- ➢ I live in a small town in the Po valley, north of Italy

What’s inside

This talk describes my experience with upstreaming the Basic eXtended CAN (bxCAN) driver you may find in the stm32fx platforms

- ➢ started as Linux kernel upstreaming
- ➢ continued with userspace upstreaming
- ➢ continuing with kernel & userspace upstreaming

This talk is **not intended** to explain:

- ➢ CAN bus
- ➢ MMU-less systems
- ➢ MMU vs MMU-less systems

Why

Feel the experience of developing and upstreaming a new driver for the Linux kernel

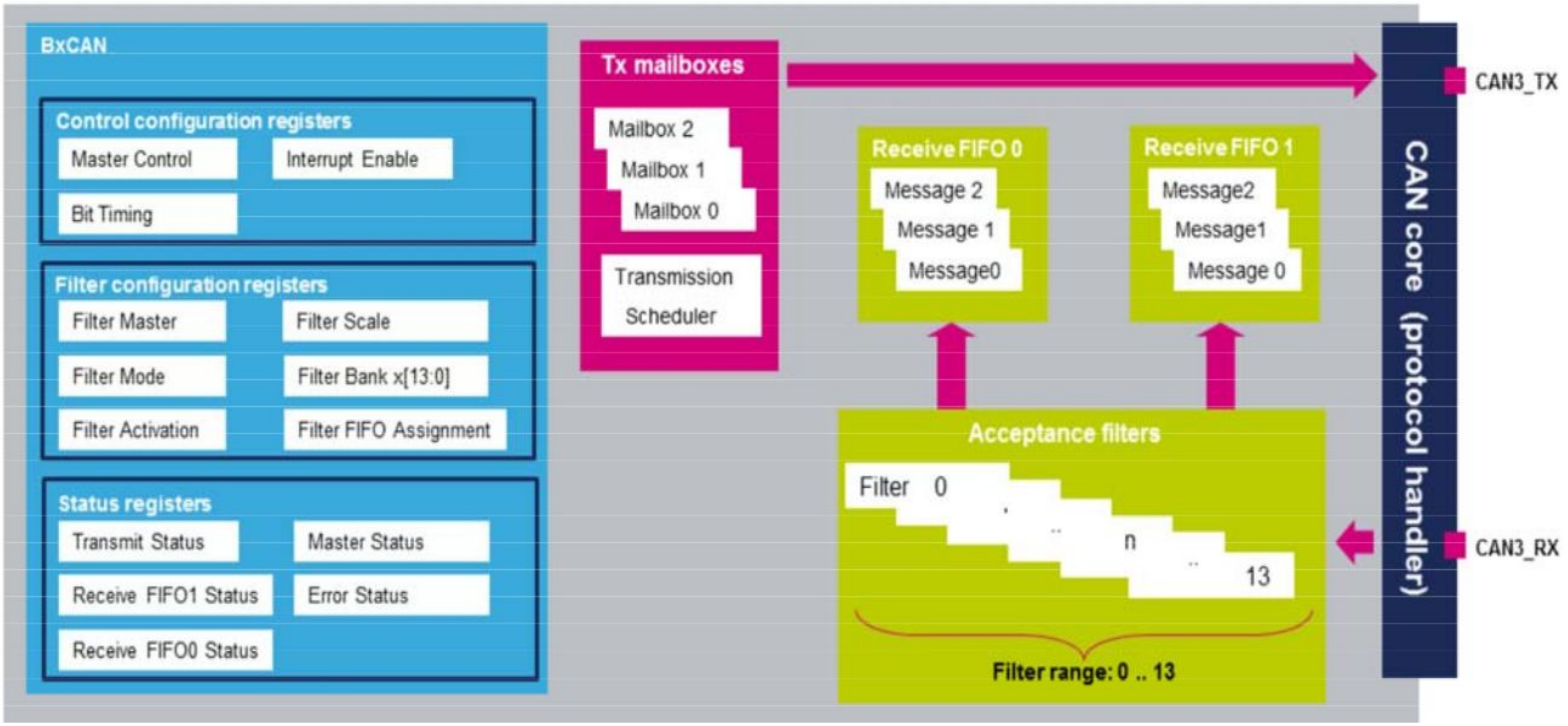
Providing answers to some curiosities of mine:

- ➢ What kind of challenges to face?
- ➢ What responsibilities come with merged code?
- ➢ What else can result from it?

Why bxCAN

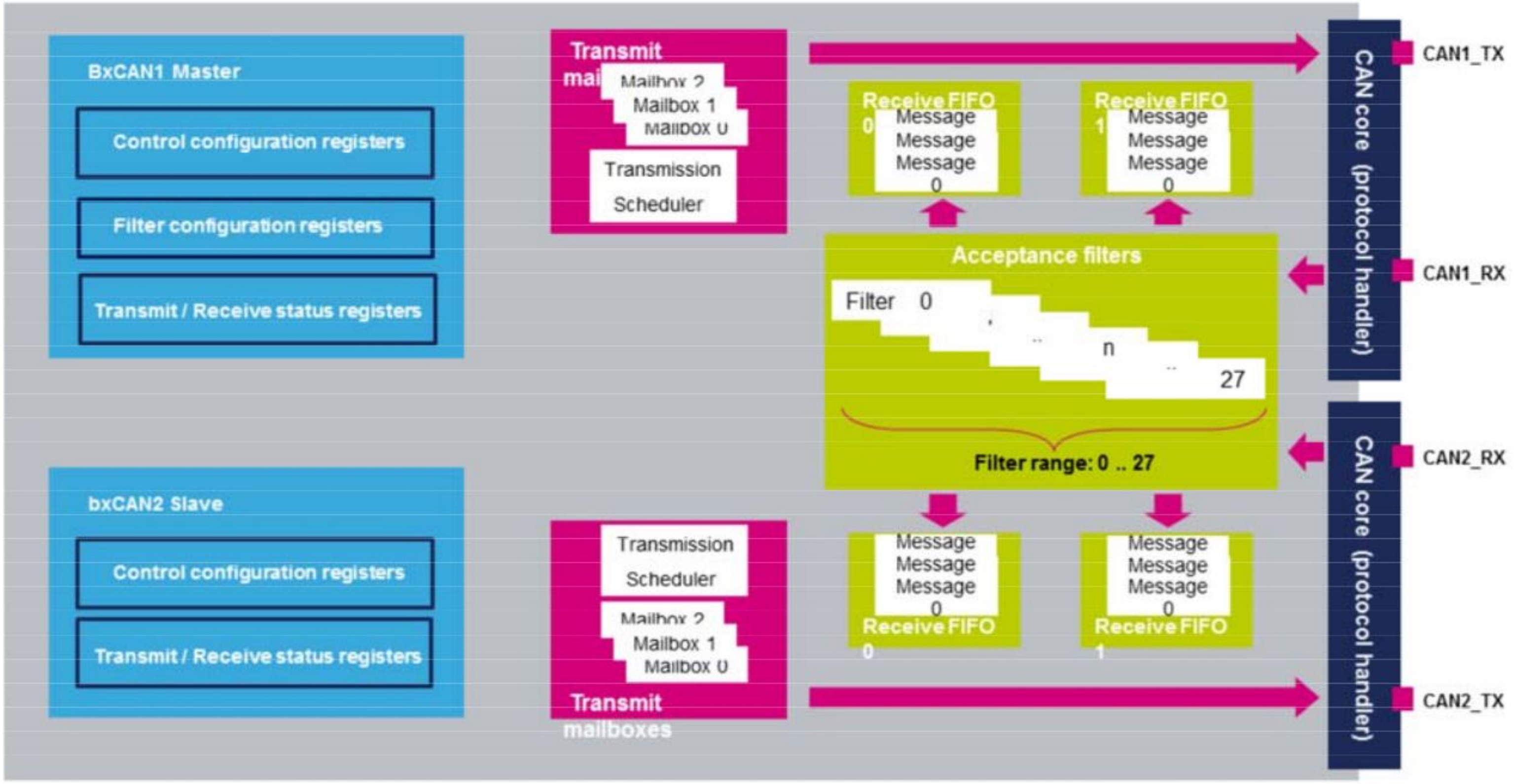
- ➢ No Linux kernel driver
- ➢ Patches upstreamed for the CAN subsystem
- ➢ The CAN subsystem maintainer and the guys are responsive and proactive
- ➢ stm32fx development boards not so expensive
- ➢ buildroot defconfig
- ➢ Zephyr driver
- ➢ Lot of code examples on-line
- ➢ Test in loopback + silent mode, no hw changes required

bxCAN - single CAN peripheral



- 14 RX filters
- Direct access to 512-byte SRAM memory

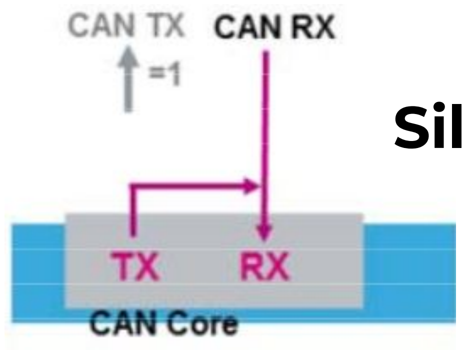
bxCAN - dual CAN peripheral



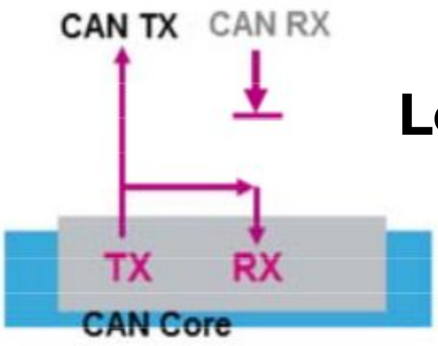
- ➢ Master/Slave -> Primary/Secondary mode
- ➢ Shared 28 RX filters
- ➢ Shared 512-byte SRAM memory
- ➢ CAN1 manage communication between CAN2 and SRAM
- ➢ CAN2 has no direct access to SRAM memory

bxCAN - test modes

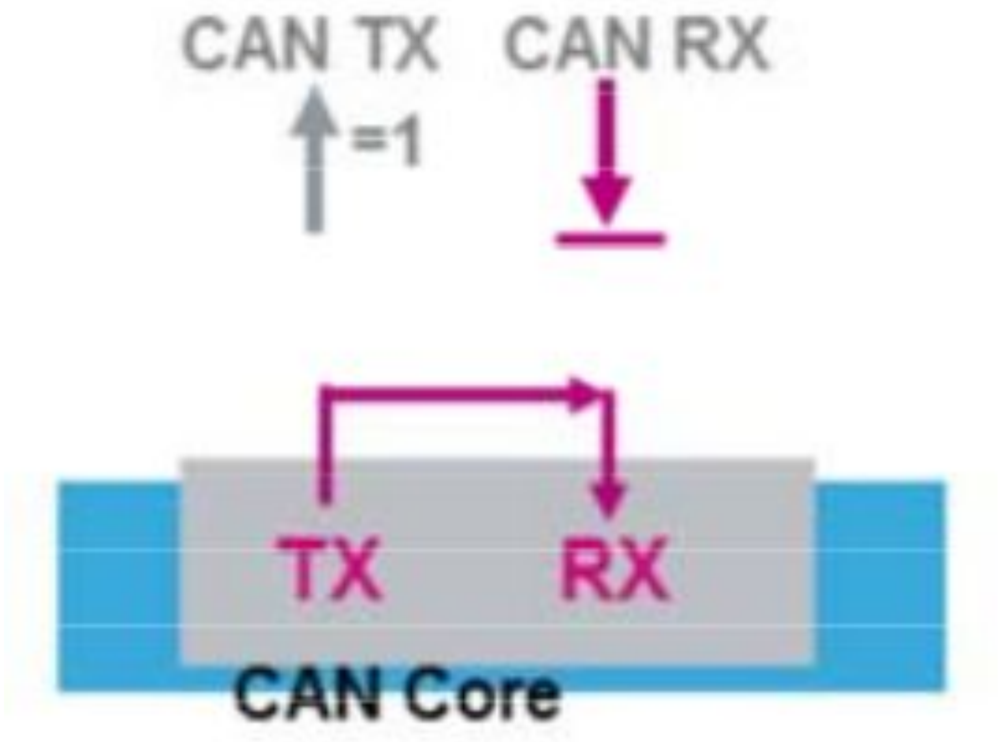
TX internally looped to RX



Silent



Loopback



host self-test

Roadmap - stm32f{4,7}69-disco

stm32f469-disco

Modify Buildroot configuration
Create Linux driver - **dual CAN**

stm32f769-disco

Create Buildroot configuration
Modify Linux driver - **single CAN**

Test the driver
Upstream the patches

stm32f469-disco - buildroot

- Use stm32f469_disco_sd_defconfig
- Enable Linux networking and CAN bus support

```
CONFIG_EPOLL=y
CONFIG_NET=y
# CONFIG_WIRELESS is not set
CONFIG_INET=y
# CONFIG_INET_DIAG is not set
# CONFIG_IPV6 is not set
CONFIG_NETDEVICES=y
CONFIG_CAN=y
```

- Set Linux override for driver development

```
LINUX_OVERRIDE_SRCDIR = ../linux-bxcan
```

- **Ready to start the driver implementation**

stm32f469-disco - driver **implementation**

- Documentation

```
properties:
  compatible:
    enum:
      - st,stm32f4-bxcan

  st,can-primary:
    description:
      Primary and secondary mode of the bxCAN peripheral is only relevant
      if the chip has two CAN peripherals. In that case they share some
      of the required logic.
      To avoid misunderstandings, it should be noted that ST documentation
      uses the terms master/slave instead of primary/secondary.
    type: boolean

  interrupts:
    items:
      - description: transmit interrupt
      - description: FIFO 0 receive interrupt
      - description: FIFO 1 receive interrupt
      - description: status change error interrupt

  st,gcan:
    $ref: /schemas/types.yaml#/definitions/phandle-array
    description:
      The phandle to the gcan node which allows to access the 512-bytes
      SRAM memory shared by the two bxCAN cells (CAN1 primary and CAN2
      secondary) in dual CAN peripheral configuration.
```

Documentation/devicetree/bindings/net/can/st,stm32-bxcan.yaml

stm32f469-disco - driver **implementation**

➢ Device tree platform

```
can1: can@40006400 {
        compatible = "st,stm32f4-bxcan";
        reg = <0x40006400 0x200>;
        interrupts = <19>, <20>, <21>, <22>;
        interrupt-names = "tx", "rx0", "rx1", "sce";
        resets = <&rcc STM32F4_APB1_RESET(CAN1)>;
        clocks = <&rcc 0 STM32F4_APB1_CLOCK(CAN1)>;
        st,can-primary;
        st,gcan = <&gcan>;
        status = "disabled";
};

gcan: gcan@40006600 {
        compatible = "st,stm32f4-gcan", "syscon";
        reg = <0x40006600 0x200>;
        clocks = <&rcc 0 STM32F4_APB1_CLOCK(CAN1)>;
};

can2: can@40006800 {
        compatible = "st,stm32f4-bxcan";
        reg = <0x40006800 0x200>;
        interrupts = <63>, <64>, <65>, <66>;
        interrupt-names = "tx", "rx0", "rx1", "sce";
        resets = <&rcc STM32F4_APB1_RESET(CAN2)>;
        clocks = <&rcc 0 STM32F4_APB1_CLOCK(CAN2)>;
        st,gcan = <&gcan>;
        status = "disabled";
};
```

arch/arm/boot/dts/st/stm32f429.dtsi

stm32f469-disco - driver **implementation**

- ➢ Device tree pin map

```
can1_pins_a: can1-0 {
	pins1 {
		pinmux = <STM32_PINMUX('B', 9, AF9)>; /* CAN1_TX */
	};
	pins2 {
		pinmux = <STM32_PINMUX('B', 8, AF9)>; /* CAN1_RX */
		bias-pull-up;
	};
};

can2_pins_b: can2-1 {
	pins1 {
		pinmux = <STM32_PINMUX('B', 13, AF9)>; /* CAN2_TX */
	};
	pins2 {
		pinmux = <STM32_PINMUX('B', 12, AF9)>; /* CAN2_RX */
		bias-pull-up;
	};
};
```

arch/arm/boot/dts/st/stm32f4-pinctrl.dtsi

stm32f469-disco - driver **implementation**

- Device tree board

```
&can1 {
	pinctrl-0 = <&can1_pins_a>;
	pinctrl-names = "default";
	status = "okay";
};

&can2 {
	pinctrl-0 = <&can2_pins_b>;
	pinctrl-names = "default";
	status = "okay";
};
```

```
&ltdc {
	status = "disabled";
};
```

Watchout!
Disable peripherals sharing pins with CAN nodes

arch/arm/boot/dts/st/stm32f469-disco.dts

stm32f469-disco - driver **implementation**

- Supported features
  - TX: all 3 available mailboxes
  - RX: FIFO 0
  - Acceptance filters
    - All incoming messages accepted
    - Filter 0 assigned to CAN1 - primary
    - Filter 14 assigned to CAN2 - secondary
    - Identifier mask mode with 32 bits width
  - Interrupts
    - TX
    - RX FIFO 0
    - Error and status change

stm32f469-disco - driver **testing**

➢ Driver probing

```
[    1.318096] CAN device driver interface
[    1.342422] bxcan 40006400.can: clk: 45000000 Hz, IRQs: 49, 48, 50
[    1.369342] bxcan 40006800.can: clk: 45000000 Hz, IRQs: 52, 51, 53
```

stm32f469-disco - driver **testing**

➢ Device communication

```
ip link set <dev> can bitrate <rate> loopback on listen-only on
ip link set up <dev>
candump <dev> -L &
cansend <dev> <msg>
```

Example:

```
ip link set can0 type can bitrate 125000 loopback on listen-only on
ip link set up can0
candump can0 -L &
cansend can0 300#AC.AB.AD.AE.75.49.AD.D1
```

stm32f469-disco - driver **testing**

command	package	**depends on MMU**
ip link	iproute2	**yes**
ip	**busybox**	no
candump	**can-utils**	**yes**
cansend	**can-utils**	**yes**

- stm32fx are MMU-less machines
- **fork()** doesn’t work on MMU-less machines
- Patching iproute2 or busybox?
  - iproute2: replace fork() with vfork()
    - What are the consequences?
  - busybox: is already in use
  - busybox: add ‘link’ sub-command
    - easier to avoid regressions
  - Can libmnl be a further option?

stm32f469-disco - driver **testing**

- Buildroot
  - Enable can-utils package
    BR2_PACKAGE_CAN_UTILS=y
  - Set override for packages development
    BUSYBOX_OVERRIDE_SRCDIR = ../busybox
    CAN_UTILS_OVERRIDE_SRCDIR = ../can-utils
- Patching **busybox**
  - Add sub-command configuration
    CONFIG_FEATURE_IP_LINK_CAN
  - **Add iplinkcan applet**
- Patching **can-utils**
  - **Don’t compile program using fork()** on MMU-less systems (bcmserver, canlogserver, isotpserver)

stm32f469-disco - driver **testing**

```
~ # ip link set can1 type can bitrate 125000 loopback on listen-only on
~ # ip link set up can1
~ # candump can1 -L &
[1] 43 candump can1 -L
~ # cansend can1 300#AC.AB.AD.AE.75.49.AD.D1
~ # (0946684872.519370) can1 300#ACABADAE7549ADD1
(0946684872.519687) can1 300#ACABADAE7549ADD1


~ # ip link set can0 type can bitrate 1000000 loopback on listen-only on
~ # ip link set up can0
~ # candump can0 -L &
[2] 49 candump can0 -L
~ # cansend can0 319#BD.BC.BE.BF.86.5A.BE.E2
~ # (0946685314.106318) can0 319#BDBCBEBF865ABEE2
(0946685314.106636) can0 319#BDBCBEBF865ABEE2
```

stm32f469-disco - driver **code review**

- Documentation
  - st,stm32-bxcan -> st,stm32f4-bxcan
  - master -> st,master
  - Drop the status property from the example
  - Use the "**syscon**" node for the **shared memory** and clocks
  - Replace the **master/slave** terms with **primary/secondary**
    - https://docs.kernel.org/process/coding-style.html under the "Naming" section
- Device tree
  - keep nodes ordered by address

stm32f469-disco - driver **code review**

- Source code
  - Fix sparse errors (make C=1)
  - Fix checkpatch warnings
  - Use **FIELD_GET()/FIELD_PREP()** macros

    ```
    cf->can_id = id >> BXCAN_RIxR_EXID_SHIFT;
    cf->can_id = FIELD_GET(BXCAN_RIxR_EXID_MASK, id);
    ```
  - Use 1 space, instead of tabs, in the macros definition

    ```
    #define BXCAN_FMR_CANSB_MASK GENMASK(13, 8)
    ```
  - Drop macros of unused peripheral registers
  - Use **regmap** functions to access **shared memory**
  - Replace the master/slave terms with primary/secondary

stm32f769-disco - buildroot

Create stm32f769_disco_sd_defconfig

- FLASH
  - U-Boot 2023.04 at 0x08000000
- SD:
  - device tree - stm32f769-disco.dtb
  - Linux 5.15.108 - zImage
  - rootfs - rootfs.ext2

stm32f769-disco - driver **implementation**

➢ Documentation

```
st,can-primary:
  description:
    Primary mode of the bxCAN peripheral is only relevant if the chip has
    two CAN peripherals in dual CAN configuration. In that case they share
    some of the required logic.
    Not to be used if the peripheral is in single CAN configuration.
    To avoid misunderstandings, it should be noted that ST documentation
    uses the terms master instead of primary.
  type: boolean

st,can-secondary:
  description:
    Secondary mode of the bxCAN peripheral is only relevant if the chip
    has two CAN peripherals in dual CAN configuration. In that case they
    share some of the required logic.
    Not to be used if the peripheral is in single CAN configuration.
    To avoid misunderstandings, it should be noted that ST documentation
    uses the terms slave instead of secondary.
  type: boolean
```

Documentation/devicetree/bindings/net/can/st,stm32-bxcan.yaml

stm32f769-disco - driver **implementation**

➢ Device tree platform

```
can3: can@40003400 {
        compatible = "st,stm32f4-bxcan";
        reg = <0x40003400 0x200>;
        interrupts = <104>, <105>, <106>, <107>;
        interrupt-names = "tx", "rx0", "rx1", "sce";
        resets = <&rcc STM32F7_APB1_RESET(CAN3)>;
        clocks = <&rcc 0 STM32F7_APB1_CLOCK(CAN3)>;
        st,gcan = <&gcan3>;
        status = "disabled";
};

gcan3: gcan@40003600 {
        compatible = "st,stm32f4-gcan", "syscon";
        reg = <0x40003600 0x200>;
        clocks = <&rcc 0 STM32F7_APB1_CLOCK(CAN3)>;
};
```

Single CAN

```
can1: can@40006400 {
        compatible = "st,stm32f4-bxcan";
        reg = <0x40006400 0x200>;
        interrupts = <19>, <20>, <21>, <22>;
        interrupt-names = "tx", "rx0", "rx1", "sce";
        resets = <&rcc STM32F7_APB1_RESET(CAN1)>;
        clocks = <&rcc 0 STM32F7_APB1_CLOCK(CAN1)>;
        st,can-primary;
        st,gcan = <&gcan1>;
        status = "disabled";
};

gcan1: gcan@40006600 {
        compatible = "st,stm32f4-gcan", "syscon";
        reg = <0x40006600 0x200>;
        clocks = <&rcc 0 STM32F7_APB1_CLOCK(CAN1)>;
};

can2: can@40006800 {
        compatible = "st,stm32f4-bxcan";
        reg = <0x40006800 0x200>;
        interrupts = <63>, <64>, <65>, <66>;
        interrupt-names = "tx", "rx0", "rx1", "sce";
        resets = <&rcc STM32F7_APB1_RESET(CAN2)>;
        clocks = <&rcc 0 STM32F7_APB1_CLOCK(CAN2)>;
        st,can-secondary;
        st,gcan = <&gcan1>;
        status = "disabled";
};
```

Dual CAN

arch/arm/boot/dts/st/stm32f746.dtsi

stm32f769-disco - driver **implementation**

- ➢ Device tree board

```
&cec {
	status = "disabled";
};

&usbotg_hs {
	status = "disabled";
};
```

Watchout!
Prevent conflicts on shared pins

```
&can1 {
	pinctrl-0 = <&can1_pins_a>;
	pinctrl-names = "default";
	status = "okay";
};

&can2 {
	pinctrl-0 = <&can2_pins_b>;
	pinctrl-names = "default";
	status = "okay";
};

&can3 {
	pinctrl-0 = <&can3_pins_a>;
	pinctrl-names = "default";
	status = "okay";
};
```

arch/arm/boot/dts/st/stm32f769-disco.dts

stm32f769-disco - driver **implementation**

- Source code
  - Support single peripheral configuration

```
drivers/net/can/bxcan.c | 34 +++++++++++++++++++++++-----------
 1 file changed, 23 insertions(+), 11 deletions(-)

+#define BXCAN_FILTER_ID(cfg) ((cfg) == BXCAN_CFG_DUAL_SECONDARY ? 14 : 0)

+enum bxcan_cfg {
+	BXCAN_CFG_SINGLE = 0,
+	BXCAN_CFG_DUAL_PRIMARY,
+	BXCAN_CFG_DUAL_SECONDARY
+};

+	if (of_property_read_bool(np, "st,can-primary"))
+		cfg = BXCAN_CFG_DUAL_PRIMARY;
+	else if (of_property_read_bool(np, "st,can-secondary"))
+		cfg = BXCAN_CFG_DUAL_SECONDARY;
+	else
+		cfg = BXCAN_CFG_SINGLE;
```

stm32f769-disco - driver **testing**

➢ Driver probing

```
[    0.549062] CAN device driver interface
[    0.557317] bxcan 40003400.can: clk: 50000000 Hz, IRQs: 33, 32, 34
[    0.568203] bxcan 40006400.can: clk: 50000000 Hz, IRQs: 36, 35, 37
[    0.579074] bxcan 40006800.can: clk: 50000000 Hz, IRQs: 39, 38, 40
```

stm32f769-disco - driver **testing**

```
~ # ip link set can0 type can bitrate 125000 loopback on listen-only on
~ # ip link set up can0
~ # candump can0 -L &
[1] 44 candump can0 -L
~ # cansend can0 300#AC.AB.AD.AE.75.49.AD.D1
~ # (0946686046.606653) can0 300#ACABADAE7549ADD1
(0946686046.606828) can0 300#ACABADAE7549ADD1

~ # ip link set can1 type can bitrate 500000 loopback on listen-only on
~ # ip link set up can1
~ # candump can1 -L &
[2] 48 candump can1 -L
~ # cansend can1 319#BD.BC.BE.BF.86.5A.BE.E2
~ # (0946686154.675659) can1 319#BDBCBEBF865ABEE2
(0946686154.675814) can1 319#BDBCBEBF865ABEE2

~ # ip link set can2 type can bitrate 1000000 loopback on listen-only on
~ # ip link set up can2
~ # candump can2 -L &
[3] 52 candump can2 -L
~ # cansend can2 324#CE.CD.CF.C0.97.6B.CF.F3
(0946686327.245944) can2 324#CECDCFC0976BCFF3
(0946686327.246106) can2 324#CECDCFC0976BCFF3
```

stm32f769-disco - driver **code review**

- ➢ Use a **syscon** node for single CAN too
- ➢ Add “st,can-secondary” to dual CAN
  - ○ st,can-primary - Dual CAN primary channel
  - ○ st,can-secondary - Dual CAN secondary channel
  - ○ **otherwise** - Single CAN
  - ○ Update not backward compatible
  - ○ No problem, the dual CAN DTS wasn’t in a stable release yet

stm32f769-disco - Azz! merge issue

- ➢ 1 series, 3 maintainers (net, mfd, platform)
- ➢ Murphy’s law …
- ➢ “… So I am afraid, this will **break** the **mainline**”

```
df362914eead ARM: dts: stm32: re-add CAN support on stm32f746            B’’
8f3ef556f8e1 dt-bindings: mfd: stm32f7: Add binding definition for CAN3  A
36a6418bb125 Revert "ARM: dts: stm32: add CAN support on stm32f746"      B’
0920ccdf41e3 ARM: dts: stm32: add CAN support on stm32f746               B
85a79b971164 can: bxcan: add support for single peripheral configuration
011644249686 ARM: dts: stm32: add pin map for CAN controller on stm32f7
6b443faa313c ARM: dts: stm32f429: put can2 in secondary mode
caf78f0f4919 dt-bindings: net: can: add "st,can-secondary" property
```

- ➢ 3 patches for 1

https://lore.kernel.org/all/20230517-corset-pelvis-5b0c41f519c9-mkl@pengutronix.de

What about the CAN tools patches?

Yes, go ahead and **upstream** your changes!

Marc

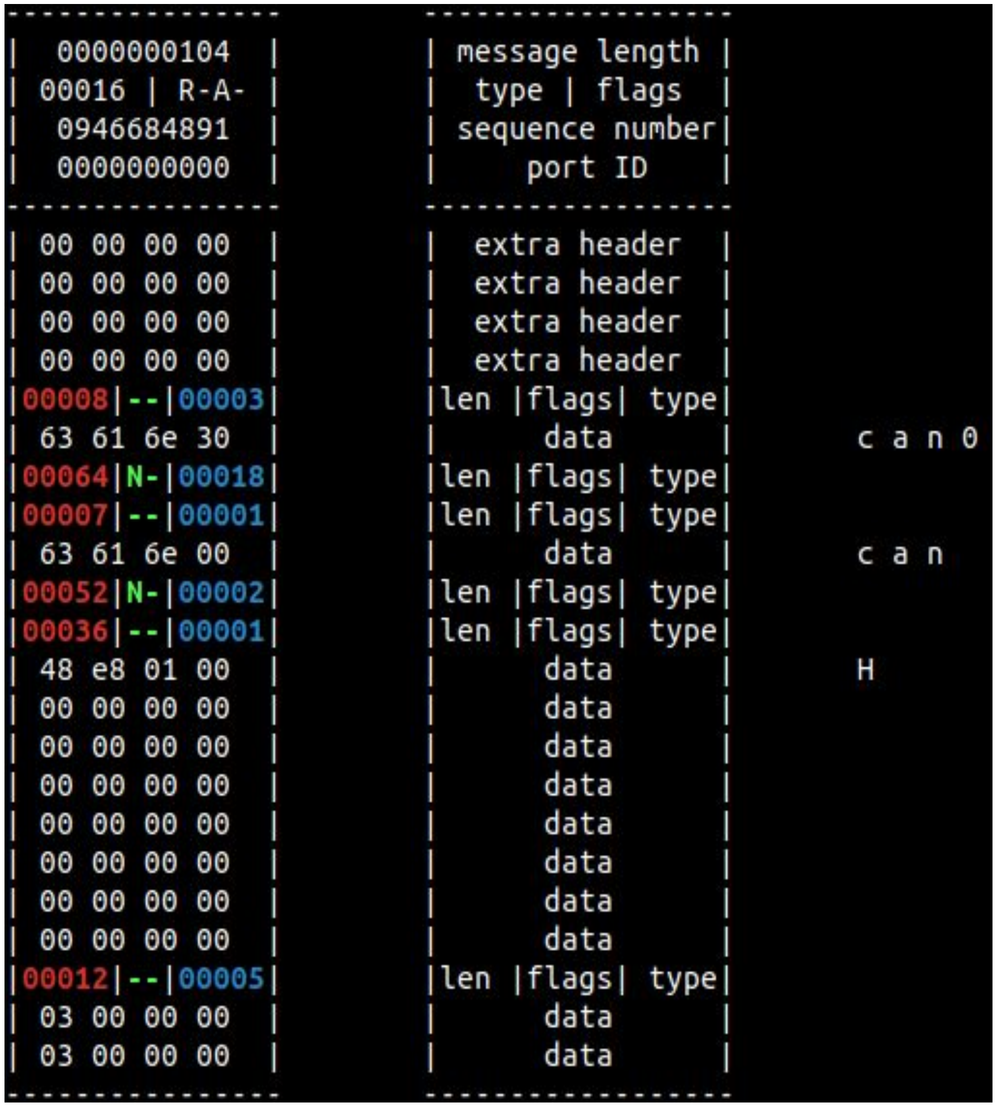
--
Pengutronix e.K. | Marc Kleine-Budde

https://lore.kernel.org/all/20230404-postage-handprint-efdb77646082@pengutronix.de

libmnl

rtnl-link-can ip link set can0 type can bitrate 125000 loopback on listen-only on

- ➢ lightweight library
- ➢ develop rtnl-link-can under examples/rtnl
- ➢ mnl_nlmsg_fprintf(), useful for debugging
- ➢ rtnl-link-set too
  rtnl-link-set <can-dev> up



```
----------------          ------------------
|  0000000104  |          | message length |
| 00016 | R-A- |          |  type | flags  |
|  0946684891  |          | sequence number|
|  0000000000  |          |    port ID     |
----------------          ------------------
| 00 00 00 00  |          |  extra header  |
| 00 00 00 00  |          |  extra header  |
| 00 00 00 00  |          |  extra header  |
| 00 00 00 00  |          |  extra header  |
|00008|--|00003|          |len |flags| type|
| 63 61 6e 30  |          |      data      |      c a n 0
|00064|N-|00018|          |len |flags| type|
|00007|--|00001|          |len |flags| type|
| 63 61 6e 00  |          |      data      |      c a n
|00052|N-|00002|          |len |flags| type|
|00036|--|00001|          |len |flags| type|
| 48 e8 01 00  |          |      data      |      H
| 00 00 00 00  |          |      data      |
| 00 00 00 00  |          |      data      |
| 00 00 00 00  |          |      data      |
| 00 00 00 00  |          |      data      |
| 00 00 00 00  |          |      data      |
| 00 00 00 00  |          |      data      |
| 00 00 00 00  |          |      data      |
|00012|--|00005|          |len |flags| type|
| 03 00 00 00  |          |      data      |
| 03 00 00 00  |          |      data      |
----------------          ------------------
```

Upstream - Linux kernel

- Dual CAN
  - can: bxcan: add support for ST bxCAN controller
  - ARM: dts: stm32: add pin map for CAN controller on stm32f4
  - ARM: dts: stm32: add CAN support on stm32f429
  - dt-bindings: net: can: add STM32 bxcan DT bindings
  - dt-bindings: arm: stm32: add compatible for syscon gcan node
- Single CAN
  - ARM: dts: stm32: re-add CAN support on stm32f746
  - dt-bindings: mfd: stm32f7: Add binding definition for CAN3
  - ARM: dts: stm32: add CAN support on stm32f746
  - can: bxcan: add support for single peripheral configuration
  - dts: stm32: add pin map for CAN controller on stm32f7
  - ARM: dts: stm32f429: put can2 in secondary mode
  - dt-bindings: net: can: add "st,can-secondary" property

Upstream - test tools

- busybox
  - ip link: support for the CAN netlink
- can-utils
  - Don't compile programs using fork() on MMU-less systems
- libmnl
  - include: cache copy of can.h and can/netlink.h
  - examples: update .gitignore files
  - examples: add rtnl-link-can

Upstream - buildroot

- package/libmnl: remove dependency on kernel headers version
- package/libmnl: fix build failure
- configs/stm32f769_disco_sd_defconfig: new defconfig
- package/libmnl: simplify LIBMNL_EXAMPLES_INSTALL_TARGETS setting
- package/libmnl: add rtnl-link-can example
- package/can-utils: enable compilation on MMU-less systems
- package/can-utils: bump to version 2023.03

Upstream

All patches were accepted except for **busybox**

If you think it could be useful for busybox to support the setup of a CAN interface, **please review the patch**

Hands-on

```
git clone -b stm32f-CAN https://github.com/passgat/buildroot.git
cd buildroot

# stm32f469
make O=output-stm32f469 stm32f469_disco_sd_defconfig
make O=output-stm32f469
# flashing U-Boot
./board/stmicroelectronics/stm32f469-disco/flash_sd.sh output-stm32f469
# dump image on SD card
dd if=output-stm32f469/images/sdcard.img of=/dev/<your-sd-device>

# stm32f769
make O=output-stm32f769 stm32f769_disco_sd_defconfig
make O=output-stm32f769
# flashing U-Boot
./board/stmicroelectronics/stm32f769-disco/flash_sd.sh output-stm32f769
dd if=output-stm32f769/images/sdcard.img of=/dev/<your-sd-device>
```

Acknowledgements

This talk was made possible thanks to the help and encouragement of many people.

First of all, thanks to all guys who contributed to the review and acceptance of the patches:
Marc Kleine-Budde, Krzysztof Kozlowski, Rob Herring, Vincent Mailhol, Lee Jones, Alexandre Torgue, Simon Horman, Stephen Rothwell, Jakub Kicinski, Paolo Abeni and the kernel test robot :) for Linux kernel, Marc Klein-Budde for can-utils, Nikolaus Voss and Bernhard Reutner-Fischer for busybox, Florian Westphal and Pablo Neira Ayuso for libmnl and Giulio Benetti, Yann E. MORIN and Thoma Petazzoni for Buildroot.

Thanks to Michael Trimarchi for coming up with the idea for this talk. Thanks to Alberto Panizzo, Andrea Ricchi, Michael Trimarchi, Vera Binacchi (my daughter) and Franco Ferrari for the slides review and refinement.

Thanks to Amarula and FOSDEM organization for the assistance and support.

I’m sorry If I forgot someone :).

Resources

STM32F7 - bxCAN
Basic Extended Controller Area Network interface
https://www.st.com/resource/en/product_training/STM32F7_Peripheral_bxCAN.pdf

STM32F4 Series
https://www.st.com/en/microcontrollers-microprocessors/stm32f4-series.html

STM32F7 Series
https://www.st.com/en/microcontrollers-microprocessors/stm32f7-series.html

789 KB Linux Without MMU on RISC-V
https://popovicu.com/posts/789-kb-linux-without-mmu-riscv

Shrinking the kernel with an axe
https://lwn.net/Articles/746780

Shrinking the kernel with a hammer
https://lwn.net/Articles/748198

Resources

Shrinking the kernel with link-time optimization
https://lwn.net/Articles/744507

Shrinking the kernel with link-time garbage collection
https://lwn.net/Articles/741494

Build Linux for STM32F769I DISCO Using Buildroot
https://adrianalin.gitlab.io/popsblog.me/posts/build-linux-for-stm32f769i-disco-using-buildroot

Linux driver upstreaming
STM32F4 - Dual CAN
https://lore.kernel.org/all/20220817143529.257908-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20220820082936.686924-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20220828133329.793324-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20220925175209.1528960-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20221017164231.4192699-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20230109182356.141849-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20230116175152.2839455-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20230326160325.3771891-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20230327201630.3874028-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20230328073328.3949796-1-dario.binacchi@amarulasolutions.com

Resources

Linux driver upstreaming
STM32F7 - Single CAN
https://lore.kernel.org/all/20230423172528.1398158-1-dario.binacchi@amarulasolutions.com
https://lore.kernel.org/all/20230427204540.3126234-1-dario.binacchi@amarulasolutions.com

can-utils upstreaming
https://github.com/linux-can/can-utils/pull/426

busybox upstreaming
http://lists.busybox.net/pipermail/busybox/2023-April/090294.html

libmnl upstreaming
https://patchwork.ozlabs.org/project/netfilter-devel/patch/20230420192115.1953830-1-dario.binacchi@amarulasolutions.com
https://patchwork.ozlabs.org/project/netfilter-devel/patch/20230520174435.3925314-1-dario.binacchi@amarulasolutions.com
https://patchwork.ozlabs.org/project/netfilter-devel/patch/20230520203512.3940990-1-dario.binacchi@amarulasolutions.com

Resources

buildroot upstreaming

https://patchwork.ozlabs.org/project/buildroot/patch/20230422140018.616018-1-dario.binacchi@amarulasolutions.com
https://patchwork.ozlabs.org/project/buildroot/patch/20230423170820.1395767-1-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230509201153.2413972-2-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230509201153.2413972-3-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230509201153.2413972-4-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230509201153.2413972-5-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230520210153.3944842-1-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230523173142.2003235-1-dario.binacchi@amarulasolutions.com/
https://patchwork.ozlabs.org/project/buildroot/patch/20230602085741.3338373-1-dario.binacchi@amarulasolutions.com/

Q/A

Thanks for your time

Questions?

Suggestions?

Comments?