UART Development Guide

Release Version：1.1

Author E-mail：hhb@rock-chips.com

Release Date：2019.01

Classified Level：Publicity

Preface

Overview

Chipset Version

Chipset	Kernel Version
All Rockchip chipset with Linux 4.4	Linux4.4
All Rockchip chipset with Linux 4.19	Linux4.19

Intended Audience

This document (this guide) is primarily intended for the following engineers:
Field Application Engineer
Software Engineer

Revision History

Date	Version	Author	Revision Introduction
2017-12-21	V1.0	Huibin.Hong	Initial Release
2019-02-14	V1.1	Huibin.Hong	Update
2019-11-13	V1.2	Huibin.Hong	Support Linux4.19

---

1 Rockchip UART Feature

UART（Universal Asynchronous Receiver/Transmitter)

The features supported by the Linux 4.4 uart driver:

- Supports up to 4M baud rate
- Some serial ports support hardware automatic flow control, some don't support. See the data sheet for details.
- Support interrupt transfer mode and DMA transfer mode

2 Linux Software

2.1 Source Code Path

The type is 16550A, which is compatible with 8250 driver.

1. drivers/tty/serial/8250/8250_core.c
2. drivers/tty/serial/8250/8250_dma.c   dma transfer code
3. drivers/tty/serial/8250/8250_dw.c   relevant to design ware ip
4. drivers/tty/serial/8250/8250_early.c   early console code
5. drivers/tty/serial/8250/8250_fsl.c
6. drivers/tty/serial/8250/8250.c
7. drivers/tty/serial/8250/8250_port.c   port related interface
8. drivers/tty/serial/earlycon.c   support early console for printk

2.2 Kernel Configuration

```
Device Drivers  --->
    Character devices  --->
        Serial drivers  --->
          [*] 8250/16550 and compatible serial support
          [ ]   Support 8250_core.* kernel options (DEPRECATED)
          [*]   Console on 8250/16550 and compatible serial port   //enable
console deiver
          [ ]   DMA support for 16550 compatible UART controllers
          (5)   Maximum number of 8250/16550 serial ports   //larger than soc
uart num
          (5)   Number of 8250/16550 serial ports to register at runtime
//larger than soc uart num
          [ ]   Extended 8250/16550 serial driver options
          [*] Support for Synopsys DesignWare 8250 quirks
```

2.3 Enable UART Device

2.3.1 Enable DTS uart0

Add the following code to the board-level DTS file:

```
&uart0 {
        status = "okay";
};
```

2.3.2 Device drive registration log

```
[0.464875] Serial: 8250/16550 driver, 5 ports, IRQ sharing disabled
[0.466561] ff180000.serial: ttyS0 at MMIO 0xff180000 (irq = 36, base_baud =
1500000) is a 16550A
[0.467112] ff1a0000.serial: ttyS2 at MMIO 0xff1a0000 (irq = 37, base_baud =
1500000) is a 16550A
[0.467702] ff370000.serial: ttyS4 at MMIO 0xff370000 (irq = 40, base_baud =
1500000) is a 16550A
```

The successful registration of the device likes the above log. If pin-ctrl conflicts with other drivers, it will report the log of pin-ctrl configuration fail.

2.3.3 Serial device

The legacy driver will register 5 ttySx devices. However, if the serial port without enable by method of Chapter 2.3.1, although there are device nodes, they cannot be operated.

```
1|root@android:/ # ls /dev/tt
ttyS0   ttyS1   ttyS2   ttyS3  ttyS4
```

If the kernel contains the following patches, the serial port driver will only generate serial ports with dts enabled.

```
commit a997ba744c6b001b8a8033aaacc65d6f4ce849a2
Author: Huibin Hong <huibin.hong@rock-chips.com>
Date:   Mon Nov 5 15:56:03 2018 +0800

    serial: 8250: add /dev/ttySx when uart is enable
    before the patch:
    ls /dev/ttyS
    ttyS0 ttyS1 ttyS2 ttyS3 ttyS4 ttyS5  ttyS6 ttyS7

    after the patch:
    ls /dev/ttyS
    ttyS3  ttyS4  ttyS6

    Change-Id: I844523408751cb579bbfb50fafb7923d5c2cafdf
```

The driver will number the serial port according to aliase, as follows: "serial0" will eventually generate ttyS0, and "serial3 "will generate ttyS3 device.

```
    aliases {
        serial0 = &uart0;
        serial1 = &uart1;
        serial2 = &uart2;
        serial3 = &uart3;
        serial4 = &uart4;
    };
```

2.4 DTS node Configuration

Take the uart0 DTS node as an example:

In the dtsi file:

```
uart0: serial@ff180000 {
        compatible = "rockchip,rk3399-uart", "snps,dw-apb-uart";
        reg = <0x0 0xff180000 0x0 0x100>;
        clocks = <&cru SCLK_UART0>, <&cru PCLK_UART0>;
        clock-names = "baudclk", "apb_pclk";
        interrupts = <GIC_SPI 99 IRQ_TYPE_LEVEL_HIGH 0>;
        reg-shift = <2>;
        reg-io-width = <4>;
        dmas = <&dmac_peri 0>, <&dmac_peri 1>;
        dma-names = "tx", "rx";
        pinctrl-names = "default";
        pinctrl-0 = <&uart0xfer &uart0cts &uart0_rts>;
        status = "disabled";
    };
```

Board level dts file added:

```
&uart0 {
        status = "okay";
};
```

2.4.1 Pin-ctrl Configuration

Sometimes a serial port has multiple sets of IOMUX configurations, which need to be configured according to the actual situation.

```
pinctrl-names = "default";
pinctrl-0 = <&uart0xfer &uart0cts &uart0_rts>;
```

Where "uart0_cts" and "uart0_rts" are hardware flow control pins, which only means that the pins are configured as corresponding function pins, and does not mean that hardware flow control is enabled. Enabling hardware flow control needs to be set from the application layer. **It should be noted that if hardware flow control is enabled, uart0_cts and uart0_rts must be matched at the same time. If it does not need hardware flow control, you can remove uart0_cts and uart0_rts.**

2.4.2 DMA transfer mode

Compared with the interrupt transfer mode, using DMA does not necessarily increase the transfer speed, but may slightly reduce the transfer speed. Because the performance of the CPU is now very high, the difficulty of improving the transmission performance is in the peripheral, otherwise, starting the DMA will consume additional resources. But overall, the interrupt mode will take up more CPU resources. Only when the amount of transmitted data is large, the mitigation effect on the CPU load will be more obvious.

Some suggestions for the use of DMA:

If the amount of data transferred by the external device is not large, please use the default interrupt mode.

If the external device transmits a large amount of data, you can use DMA.

If the serial port is not connected to the auto flow control pin, you can use DMA as a FIFO buffer to prevent data loss.

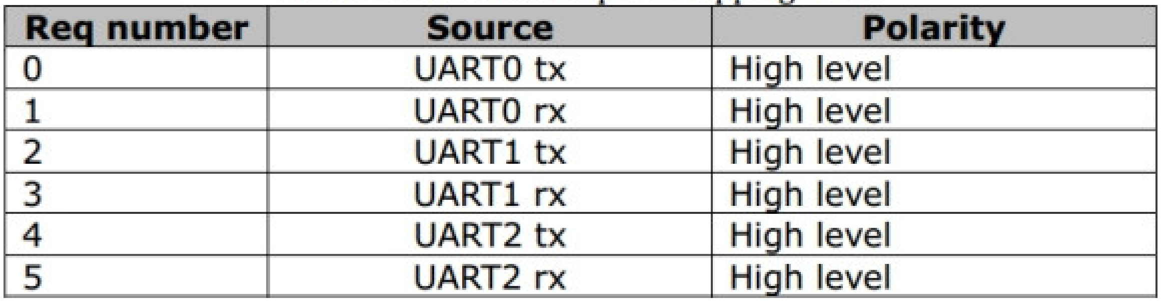
The following configuration is required when you use DMA, if there is no the configuration like below,you need to add it by yourself:

`dma-names = "tx", "rx";` Enable DMA transmission and receiving

`dma-names = "!tx", "!rx";` Disable DMA transmission and receiving

`dmas = <&dmac_peri 0>, <&dmac_peri 1>; Here" 0" and "1" are the channel numbers of the peripheral and DMAC connections. The DMAC uses this number to identify the peripherals. Find the Req number through the manual, as shown below.

Table 12-2 DMAC1 Request Mapping Table



Req number	Source	Polarity
0	UART0 tx	High level
1	UART0 rx	High level
2	UART1 tx	High level
3	UART1 rx	High level
4	UART2 tx	High level
5	UART2 rx	High level

According to the manual, determine which DMAC the peripheral belongs to, select "&dmac_peri" or "&dmac_bus". Generally, DMAC1 is" dmac_peri".

DMAC0 is dmac_bus.

As below:

```
amba {
compatible = "arm,amba-bus";
```

```
        #address-cells = <2>;
        #size-cells = <2>;
        ranges;

        dmac_bus: dma-controller@ff6d0000 {
            compatible = "arm,pl330", "arm,primecell";
            reg = <0x0 0xff6d0000 0x0 0x4000>;
            interrupts = <GIC_SPI 5 IRQ_TYPE_LEVEL_HIGH 0>,
                     <GIC_SPI 6 IRQ_TYPE_LEVEL_HIGH 0>;
            #dma-cells = <1>;
            clocks = <&cru ACLK_DMAC0_PERILP>;
            clock-names = "apb_pclk";
            peripherals-req-type-burst;
        };

        dmac_peri: dma-controller@ff6e0000 {
            compatible = "arm,pl330", "arm,primecell";
            reg = <0x0 0xff6e0000 0x0 0x4000>;
            interrupts = <GIC_SPI 7 IRQ_TYPE_LEVEL_HIGH 0>,
                     <GIC_SPI 8 IRQ_TYPE_LEVEL_HIGH 0>;
            #dma-cells = <1>;
            clocks = <&cru ACLK_DMAC1_PERILP>;
            clock-names = "apb_pclk";
            peripherals-req-type-burst;
        };
    };
```

Note:

Some scenarios that do not require DMA, you can also consider turning off DMA as sending and receiving, as follows:

```
dma-names = "!tx", "!rx";
```

And generates log below:

```
[54696.575402] ttyS0 - failed to request DMA, use interrupt mode
```

Due to the limited resources of the DMA channel, you can consider turning off the DMA transmission of TX in the case of tight channel resources, as follows:

```
dma-names = "!tx", "rx";
```

And generates log below:

```
[  498.889713] dw-apb-uart ff0a0000.serial: got rx dma channels only
```

2.4.3 Baud Rate Configuration Instructions

Baud rate = clock source / 16 / DIV. (DIV is the division factor)

The code will set the clock according to the baud rate currently. Generally, the baud rate below 1.5M can be divided. The baud rate above 1.5M may be divided by fractional or integer. If the above is not work, you need to modify the PLL. However, modifying the PLL is risky and it will affect other modules. If you need help, please post your demand through Rockchip Redmine.

If the following log occurs while operating the serial port, you need to print the clock tree to determine whether the serial port clock setting is correct.

```
[54131.273012] rockchip_fractional_approximation parent_rate(676000000) is low than
    rate(48000000)*20, fractional div is not allowed
```

The following commands must be input when the serial port is turn-on, otherwise clk may not be accurate. In this example, the serial port is set to the baud rate of 3M. It can be seen from the following log that the serial port is the clk_uart4_pmu integer frequency division, the clk close to 48M, which divided from the 676M PLL. Here 48M is the minimum clock to provide 3M baud rate，calculated by the above formula. Although there is an error, within the allowable range, the range of this error is set to plus or minus 2%.

```
root@android:/ # cat /sys/kernel/debug/clk/clk_summary | grep uart
          clk_uart4_src                   1             1   676000000          0
0
             clk_uart4_div                1             1    48285715          0
0
                clk_uart4_pmu             1             1    48285715          0
0
                clk_uart4_frac            0             0      285257          0
0
             pclk_uart4_pmu               1             1    48285715          0
0
```

2.4.4 Serial port wake-up system

For this feature, the kernel needs to be patched and the SOC trust firmware may need to be modified. If you need, please consult the one who maintain the trust firmware.

```
&uart0 {
    wakeup-source;    //Enable the serial port wake-up function, which makes the serial port active during standby, and set the serial port interrupt as the wake-up source.
    status = "okay";
};
```

3 Linux serial port print

3.1 FIQ debugger, ttyFIQ0 device as console

3.1.1 DTS enable node fiq_debugger，disable the corresponding node of uart

```
fiq_debugger: fiq-debugger {
        compatible = "rockchip,fiq-debugger";
        rockchip,serial-id = <2>;    /*Set the serial port id. If you want to change the serial port, change this ID.*/
        rockchip,wake-irq = <0>;
        rockchip,irq-mode-enable = <0>;  /* If 1， uart uses irq instead of fiq */
        rockchip,baudrate = <1500000>;  /* Only 115200 and 1500000 */
        pinctrl-names = "default";
        pinctrl-0 = <&uart2c_xfer>;     /*After changing different serial ports, 
you need to configure iomux*/
        interrupts = <GIC_SPI 150 IRQ_TYPE_LEVEL_HIGH 0>;  /* Configure signal 
irq, which can be the maximum SOC interrupt number plus one. */
        status = "okay";
};
/*Disable corresponding uart nodes*/
&uart2 {
    status = "disabled";
};
```

The node will register the" /dev/ttyFIQ0 "device after driver loading. It is notes that rockchip, serial-id, even if changed, is also registered as ttyFIQ0.

Rockchip, irq-mode-enable = <0>: If this value is "1 "and the serial port interrupt mode uses irq, generally no problem. But if it is 0 and it uses FIQ mode, and some platforms with trust firmware need to be used with caution. This may be a failure due to mismatch of trust firmware version and the kernel version.

3.1.2 Enable early printk

Add the following parameters, where 0xff1a0000 is the physical base address of uart2. And the different serial port are different base addresses.

Generally, the following parameters do not add the baud rate of 115200 additionally, the baud rate configured by U-Boot or loader can be used.

If you have configured baud rate, some bug might be occurs, because the kernel early con does not support this part very well.

```
chosen {
    bootargs ="earlycon=uart8250,mmio32,0xff1a0000";
};
```

3.1.3 Android parameter.txt configure console device

Generally, the following parameters can be specified by the default console device, such as ttyFIQ0 registered above, will be used. But if you specify ttyS2, you can't type the command.

```
commandline: androidboot.console=ttyFIQ0  console=ttyFIQ0
```

3.2 ttySx as console

3.2.1 uart2 as console

Add the following parameters, where 0xff1a0000 is the physical base address of uart2. And the different serial port are different base addresses.

Generally, the following parameters do not add the baud rate of 115200 additionally, the baud rate configured by U-Boot or loader can be used.

If you have configured baud rate, some bug might be occurs, because the kernel early con does not support this part very well.

```
chosen {
    bootargs ="console=uart8250,mmio32,0xff1a0000";
};

&uart2 {
  status = "okay";
};
```

3.2.2 Enable early printk

```
console=uart8250,mmio32,0xff1a0000  //It already contains the function of early printk
```

3.2.3 Android parameter.txt configure console device

Generally, the following parameters can be specified by the default console device, such as ttyS2 registered above, will be used. But if you specify ttyFIQ0, you can't type the command.

```
commandline: androidboot.console=ttyS2 console=ttyS2
```

Note: console from chapter 3.1 and chapter 3.2 can not exist at the same time, otherwise print error. rockchip,`serial-id = <x>` of FIQ debugger and ttySx are mutually exclusive, which means that a serial port is used by the fiq debugger driver, it can not be used as a common serial port.

3.3 Turn off the serial port printing

3.3.1 Remove or disable all configurations of chapter 3.1 or 3.2

3.3.2 Remove the configuration of the 8250 driver console

```
Device Drivers  --->
    Character devices  --->
        Serial drivers  --->
            [ ] Console on 8250/16550 and compatible serial port
```

If you do not want to modify this configuration, you need to add "console=" and specify nothing at the command line, which means that the console is not applicable.

3.3.3 Removes recovery on the console at Android device, otherwise it will get stuck as reset

```
android/device/rockchip/common/recovery/etc/init.rc
service recovery /sbin/recovery
#console  //Comment out this
seclabel u:r:recovery:s0
```

4 Debug serial device

It is better to use the tested APK software instead of command "echo cat "to debug the serial device. Or, consult Rockchip FAE to get the ts_uart test bin file. Entering ts_uart on the command line will get the relevant help information.

```
1|root@android:/ # ts_uart
 Use the following format to run the HS-UART TEST PROGRAM
 ts_uart v1.0
 For sending data:
 ./ts_uart <tx_rx(s/r)> <file_name> <baudrate> <flow_control(0/1)> <max_delay(0-
100)> <random_size(0/1)>
 tx_rx : send data from file (s) or receive data (r) to put in file
 file_name : file name to send data from or place data in
 baudrate : baud rate used for TX/RX
 flow_control : enables (1) or disables (0) Hardware flow control using RTS/CTS
lines
 max_delay : defines delay in seconds between each data burst when sending.
Choose 0 for continuous stream.
 random_size : enables (1) or disables (0) random size data bursts when sending.
Choose 0 for max size.
 max_delay and random_size are useful for sleep/wakeup over UART testing. ONLY
meaningful when sending data
 Examples:
 Sending data (no delays)
 ts_uart s init.rc 1500000 0 0 0 /dev/ttyS0
 loop back mode:
 ts_uart m init.rc 1500000 0 0 0 /dev/ttyS0
 receive then send
 ts_uart r init.rc 1500000 0 0 0 /dev/ttyS0
```

If the serial port APK cannot turn on the serial device, there may be no permission . You need to modify the device permissions of /dev/ttySx to 0666.

Take Android as an example. Add the following configuration to ueventd.rc. If it still doesn't work, please contact the Android developer to modify the permissions.

```
/dev/ttySx            0666   system      system
```