



StarFive
赛昉科技

昉·惊鸿7110 UART开发手册

昉·星光 2

版本：1.0

日期：2022/11/10

Doc ID: JH7110-DGCH-004

法律声明

阅读本文件前的重要法律告知。

版权注释

版权 ©上海赛昉科技有限公司，2023。版权所有。

本文档中的说明均基于“视为正确”提供，可能包含部分错误。内容可能因产品开发而定期更新或修订。上海赛昉科技有限公司（以下简称“赛昉科技”）保留对本协议中的任何内容进行更改的权利，恕不另行通知。

赛昉科技明确否认任何形式的担保、解释和条件，无论是明示的还是默示的，包括但不限于适销性、特定用途适用性和非侵权的担保或条件。

赛昉科技无需承担因应用或使用任何产品或电路而产生的任何责任，并明确表示无需承担任何及所有连带责任，包括但不限于间接、偶然、特殊、惩戒性或由此造成的损害。

本文件中的所有材料受版权保护，为赛昉科技所有。不得以任何方式修改、编辑或断章取义本文件中的说明，本文件或其任何部分仅限用于内部使用或教育培训。

联系我们：

地址：浦东新区盛夏路61弄张润大厦2号楼502，上海市，201203，中国

网站：<http://www.starfivetech.com>

邮箱：

- sales@starfivetech.com（销售）
- support@starfivetech.com（支持）

前言

关于本指南和技术支持信息

关于本手册

本手册主要为SDK开发和移植提供赛昉科技新一代SoC平台——昉·惊鸿7110的URAT编程基础和调试操作。

受众

本手册主要服务于与URAT相关驱动程序的开发人员。如果您正在开发其他模块，请与您的销售或支持顾问联系，获取昉·惊鸿7110的完整文档。






修订历史

表 0-1 修订历史

Version	发布说明	修订
1.0	2022/11/10	首次发布。

注释和注意事项

本指南中可能会出现以下注释和注意事项：

-  **提示：**
建议如何在某个主题或步骤中应用信息。
-  **注：**
解释某个特例或阐释一个重要的点。
-  **重要：**
指出与某个主题或步骤有关的重要信息。
-  **警告：**
表明某个操作或步骤可能会导致数据丢失、安全问题或性能问题。
-  **警告：**
表明某个操作或步骤可能导致物理伤害或硬件损坏。

目录

表格清单.....	5
插图清单.....	6
法律声明.....	ii
前言.....	iii
1. 简介.....	7
1.1. 功能层.....	7
1.2. 设备树概述.....	7
1.3. 源代码结构.....	8
2. 配置.....	9
2.1. 内核菜单配置.....	9
2.2. 设备树代码.....	11
2.3. 设备树配置.....	12
2.4. 板级配置.....	13
2.5. 将其他UART设置为打印控制台.....	14
3. 接口介绍.....	15
3.1. 启用或禁用串口.....	15
3.2. 配置串口属性.....	15
3.2.1. tcgetattr.....	15
3.2.2. tcsetattr.....	16
3.2.3. cfgetispeed.....	16
3.2.4. cfgetospeed.....	16
3.2.5. cfsetispeed.....	16
3.2.6. cfsetospeed.....	16
3.2.7. cfsetspeed.....	16
3.2.8. tcflush.....	16
4. 示例用例.....	17

表格清单

表 0-1 修订历史..... iii



插图清单

图 1-1 功能层.....	7
图 1-2 设备树 workflow.....	8
图 2-1 Device Drivers.....	9
图 2-2 Character Devices.....	10
图 2-3 Serial Drivers.....	10
图 2-4 Support for 8250.....	11



1. 简介

通用异步收发器（UART）使用一条数据传输线将数据发送到目的地。

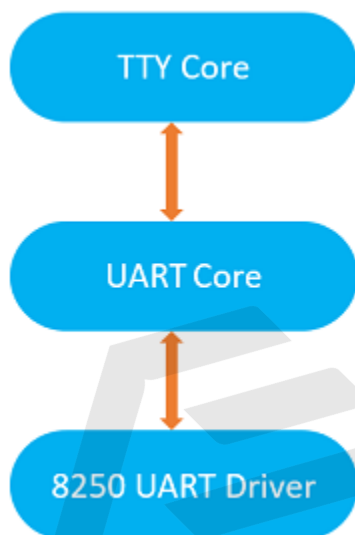
UART不仅可以输出日志数据进行系统调试，还可以完成短距离通信。它是一个在嵌入式系统中具有实际用途的接口。

1.1. 功能层

昉·惊鸿7110 SoC平台的UART驱动程序有以下几层。

- TTY核：TTY指TeleType和/或TeleType Writer，用于注册并管理核中的所有TTY设备。
- UART核：它为UART驱动程序提供了一组API，用于注册设备和驱动程序。
- 8250 UART驱动程序：它是昉·惊鸿7110 SoC平台的初始化和数据通信平台。

图 1-1 功能层



1.2. 设备树概述

自Linux 3.x以来，系统就引入了设备树作为数据结构和语言来描述硬件配置。设备树是硬件设置的系统可读描述，这样操作系统不必硬编码机器的详细信息。

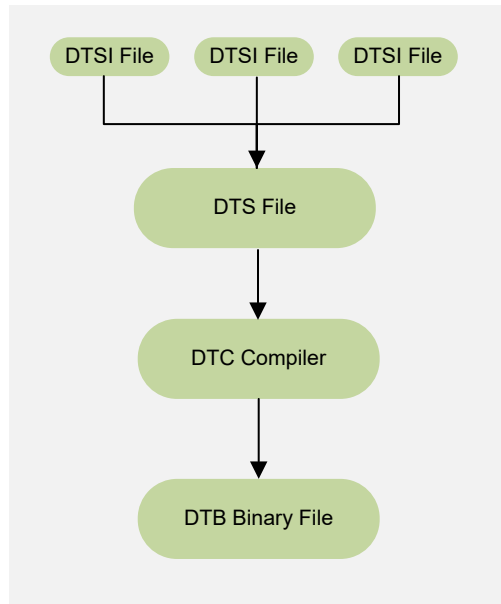
一个设备树主要有以下呈现形式。

- 设备树编译器（DTC）：用于将设备树编译为系统可读的二进制文件的工具。
- 设备树源码（DTS）：人类可读的设备树描述文件。您可以在此文件中找到目标参数并修改硬件配置。

- 设备树源码信息 (DTSI)：可包括在设备树描述中的人类可读的头文件。您可以在此文件中找到目标参数并修改硬件配置。
- 设备树块 (DTB)：系统可读设备树二进制blob文件，在系统中烧录以供执行。

下图显示了上述形式的关系（工作流）。

图 1-2 设备树工作流



1.3. 源代码结构

以下代码块为URAT驱动程序的源代码结构。

```
linux 5.15
|-- include
|   |-- linux
|   |   |-- serial_8250.h
|   |   |-- serial_core.h
|-- driver
|   |-- tty
|   |   |-- serial
|   |   |   |-- serial_core.c
|   |   |   |-- 8250
|   |   |   |   |-- 8250_dw.c
|   |   |   |   |-- 8250_core.c
|   |   |   |   |-- 8250.h
```


2. 配置

2.1. 内核菜单配置

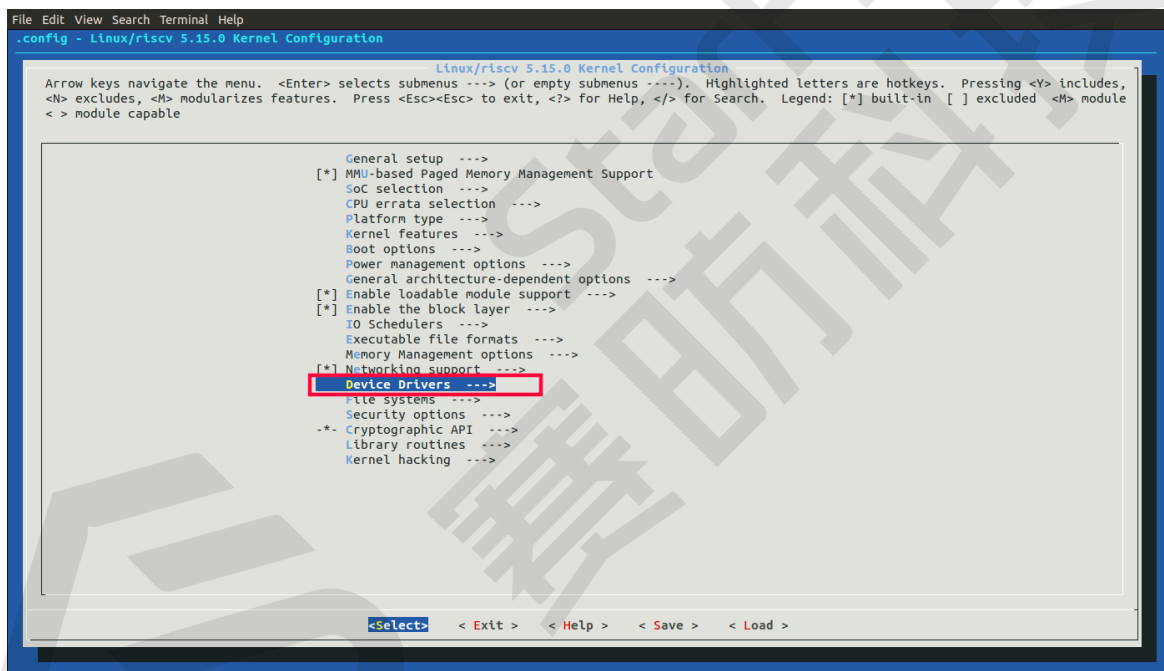
执行以下步骤，创建URAT内核配置：

1. 在freelight-u-sdk的根目录下，输入以下命令以进入内核菜单配置GUI。

```
make linux-menuconfig
```

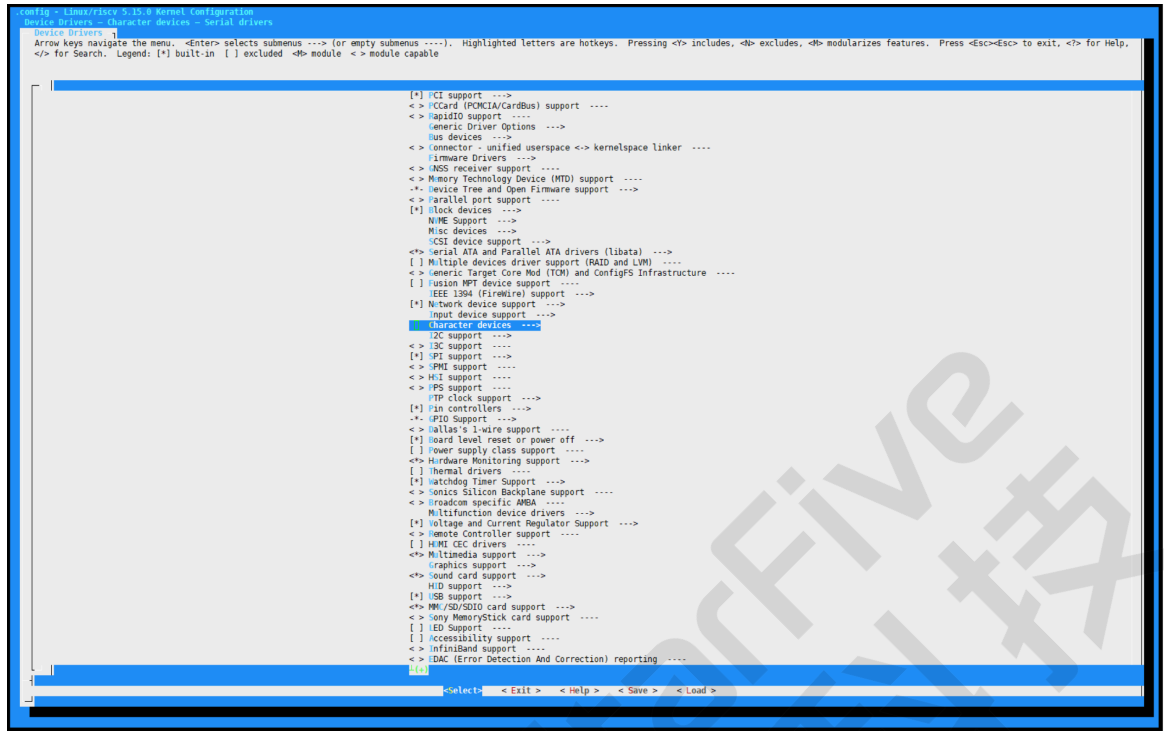
2. 进入Device Drivers菜单。

图 2-1 Device Drivers



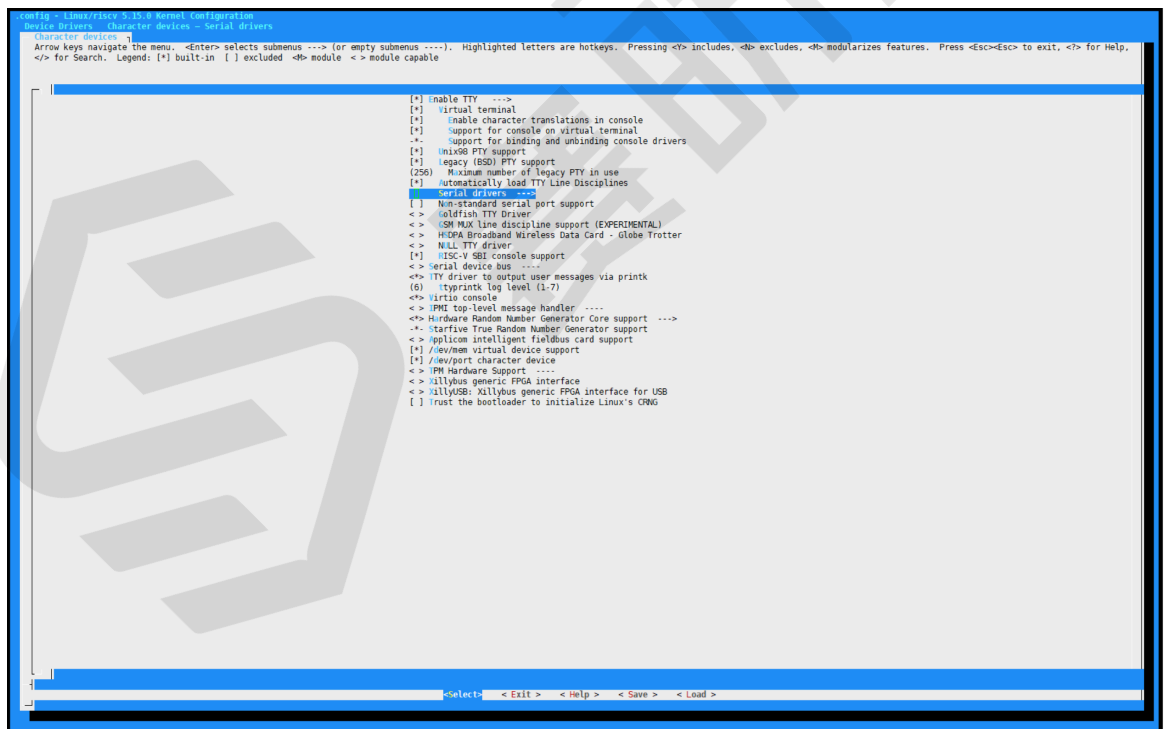
3. 进入Character devices菜单。

图 2-2 Character Devices



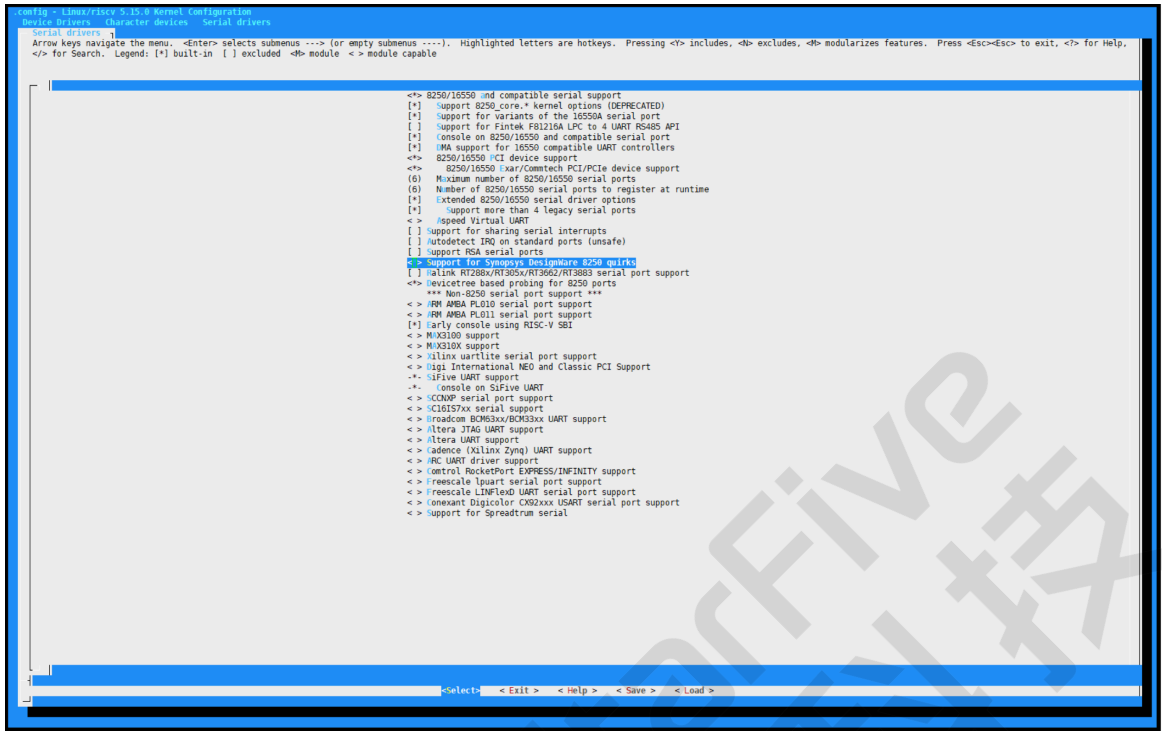
4. 进入Serial drivers菜单。

图 2-3 Serial Drivers



5. 选择Support for Synopsys Designware 8250 quirks选项。

图 2-4 Support for 8250



6. 保存更改，并退出内核配置对话框。

2.2. 设备树代码

总体结构

昉·惊鸿7110的设备树代码如下：

```

linux
├── arch
│   ├── riscv
│   │   ├── boot
│   │   │   ├── dts
│   │   │   │   └── starfive
│   │   │   │       ├── codecs
│   │   │   │       │   ├── sf_pdm.dtsi
│   │   │   │       │   ├── sf_pwm dac.dtsi
│   │   │   │       │   ├── sf_spdif.dtsi
│   │   │   │       │   ├── sf_tdm.dtsi
│   │   │   │       └── sf_wm8960.dtsi
│   │   │   └── evb-overlay
│   │   │       ├── jh7110-evb-overlay-can.dts
│   │   │       ├── jh7110-evb-overlay-rgb2hdmi.dts
│   │   │       ├── jh7110-evb-overlay-sdio.dts
│   │   │       ├── jh7110-evb-overlay-spi.dts
│   │   │       └── jh7110-evb-overlay-uart4-emmc.dts
  
```



```

resets = <&rstgen RSTN_U0_DW_UART_APB>,
        <&rstgen RSTN_U0_DW_UART_CORE>;
interrupts = <32>;
status = "disabled";
};

```

以下提供了对上述代码块中的参数说明。

- **compatible**: 兼容性信息，用于连接驱动程序和目标设备。
- **reg**: 寄存器基本地址“0x10000000”和范围“0x10000”。请确保您没有更改它之后的2位，**reg-io-width**和**reg-shift**。
- **clocks**: URAT模块使用到的时钟。
- **clock-names**: 上述时钟的名称。
- **resets**: URAT模块使用到的复位信号。
- **interrupts**: 硬件中断ID。
- **status**: URAT模块的工作状态。要启用模块，请将此位设置为“okay”；要禁用该模块，请将此位设置为“disabled”。

您可以在设备树中配置每个URAT控制器。一个UART节点表示一个UART控制器。您需要为UART节点指定一个别名（alias），以便您能够从其他节点中识别它。

2.4. 板级配置

board.dts文件用于存储板级配置文件。

对于昉·星光 2单板计算机，board.dts文件位于以下路径：

```
linux/arch/riscv/boot/dts/starfive/jh7110-visionfive-v2.dts
```

以URAT0为例，其board.dts文件位于以下路径：

```
linux/arch/riscv/boot/dts/starfive/jh7110-visionfive-v2.dts
```

在文件中，您可以找到关于UART pin控制配置的以下配置信息：

```

&gpio {
    uart0_pins: uart0-pins {
        uart0-pins-tx {
            sf,pins = <PAD_GPIO5>;
            sf,pin-ioconfig = <IO(GPIO_IE(1) | GPIO_DS(3))>;
            sf,pin-gpio-dout = <GPO_UART0_SOUT>;
            sf,pin-gpio-doen = <OEN_LOW>;
        };

        uart0-pins-rx {

```

```

        sf,pins = <PAD_GPIO6>;
        sf,pinmux = <PAD_GPIO6_FUNC_SEL 0>;
        sf,pin-ioconfig = <IO(GPIO_IE(1) | GPIO_PU(1))>;
        sf,pin-gpio-doen = <OEN_HIGH>;
        sf,pin-gpio-din = <GPI_UART0_SIN>;
    };
};
};

```

您也可以找到关于pin控制的配置信息。

```

&uart0 {
    pinctrl-names = "default";
    pinctrl-0 = <&uart0_pins>;
    status = "okay";
};

```

2.5. 将其他UART设置为打印控制台

按照以下步骤完成设置。

1. 在board.dts文件中找到目标UART端口，并确保该端口已启用。

```

&uart3 {
    pinctrl-names = "default";
    pinctrl-0 = <&uart3_pins>;
    status = "okay";
};

```

2. 修改前启动步骤传递的内核命令行参数，以使用目标UART端口作为打印控制台。

```

earlyprintk console=ttyS3,115200 debug rootwait earlycon=sbi
Note:
ttyS0 <=====> uart0
ttyS1 <=====> uart1
ttyS2 <=====> uart2
ttyS3 <=====> uart3

```

3. 接口介绍

UART驱动程序自动注册并生成该设备用于串行通信：`/dev/ttySx`。

在处理UART相关应用程序时，建议Linux应用程序开发人员遵循标准Linux编码实践。

3.1. 启用或禁用串口

启用或禁用串口时，请确保遵守以下规则：

- 包括以下所有头文件：

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
```

- 请确保使用标准函数打开和关闭所需的文件。

```
int open(const char *pathname, int flags);
int close (int fd)
```

3.2. 配置串口属性

请确保在设置属性前包含以下所有文件：

```
#include <termios.h>
```

通常，一个串口具有如下属性：

- 波特率
- 数据位
- 停止位
- 校验位
- 流量控制

以下接口用于配置串口的属性。

3.2.1. tcgetattr

`tcgetattr`是一个Linux标准接口，用于从终端获取参数。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.2. tcsetattr

tcsetattr是一个Linux标准接口，用于为终端设置参数。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.3. cfgetispeed

cfgetispeed是一个Linux标准接口，用于获取输入波特率。。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.4. cfgetospeed

cfgetospeed是一个Linux标准接口，用于获取输出波特率。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.5. cfsetispeed

cfsetispeed是一个Linux标准接口，用于设置输入波特率。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.6. cfsetospeed

cfsetospeed是一个Linux标准接口，用于设置输出波特率。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.7. cfsetspeed

cfsetspeed是一个Linux标准接口，用于设置输入和输出的速度。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

3.2.8. tcflush

tcflush是一个Linux标准接口，用于丢弃写入到该对象的数据。

参见[Linux man-pages project](#)中的*Library Functions*获取更多信息。

4. 示例用例

以下演示程序包括打开、监听UART设备以及在检测到可读数据时打印的完整过程。

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h> /*File control definition*/
#include <termios.h> /*PPSIX Terminal operating system definition*/
#include <stdio.h> /*Standard input and output definition*/
#include <unistd.h> /*UNIX standard function definition*/

#define BAUDRATE          115200
#define UART_DEVICE       "/dev/ttyS3"

#define FALSE            -1
#define TRUE              0

/**
 * @brief Set communication speed for UART
 * @param fd      Type int   Open UART file
 * @param speed   Type int   UART speed
 * @return void
 */
int speed_arr[] = {B115200, B38400, B19200, B9600, B4800, B2400, B1200,
                  B300,
                  B115200, B38400, B19200, B9600, B4800, B2400, B1200, B300, };
int name_arr[] = {115200, 38400, 19200, 9600, 4800, 2400, 1200, 300,
                 115200, 38400, 19200, 9600, 4800, 2400, 1200, 300, };
void set_speed(int fd, int speed){
    int i;
    int status;
    struct termios Opt;
    tcgetattr(fd, &Opt);
    for ( i= 0; i < sizeof(speed_arr) / sizeof(int); i++) {
        if (speed == name_arr[i]) {
            tcflush(fd, TCIOFLUSH);
            cfsetispeed(&Opt, speed_arr[i]);
            cfsetospeed(&Opt, speed_arr[i]);
            status = tcsetattr(fd, TCSANOW, &Opt);
            if (status != 0) {
                perror("tcsetattr fd1");
                return;
            }
            tcflush(fd,TCIOFLUSH);
        }
    }
}
```

```

/**
 * @brief Set data bit, stop bit and parity check bit
 * @param fd Type int OPEN UART file
 * @param databits Type int data bit Value 7 or 8
 * @param stopbits Type int stop bit Value 1 or 2
 * @param parity Type int parity check Type Value N,E,O,,S
 */
int set_Parity(int fd,int databits,int stopbits,int parity)
{
    struct termios options;
    if ( tcgetattr( fd,&options) != 0) {
        perror("SetupSerial 1");
        return(FALSE);
    }
    options.c_cflag &= ~CSIZE;
    switch (databits) /*Set dataa bit count*/
    {
        case 7:
            options.c_cflag |= CS7;
            break;
        case 8:
            options.c_cflag |= CS8;
            break;
        default:
            fprintf(stderr,"Unsupported data size\n"); return (FALSE);
    }
    switch (parity)
    {
        case 'n':
        case 'N':
            options.c_cflag &= ~PARENB; /* Clear parity enable */
            options.c_iflag &= ~INPCK; /* Enable parity checking */
            break;
        case 'o':
        case 'O':
            options.c_cflag |= (PARODD | PARENB); /* Set as odd parity check*/
            options.c_iflag |= INPCK; /* Disable parity check */
            break;
        case 'e':
        case 'E':
            options.c_cflag |= PARENB; /* Enable parity */
            options.c_cflag &= ~PARODD; /* Set as even parity check*/
            options.c_iflag |= INPCK; /* Disable parity check */
            break;
        case 'S':
        case 's': /*as no parity*/
            options.c_cflag &= ~PARENB;
            options.c_cflag &= ~CSTOPB; break;
        default:
            fprintf(stderr,"Unsupported parity\n");
    }
}

```

```

    return (FALSE);
}
/* Set stop bit*/
switch (stopbits)
{
    case 1:
        options.c_cflag &= ~CSTOPB;
        break;
    case 2:
        options.c_cflag |= CSTOPB;
        break;
    default:
        fprintf(stderr, "Unsupported stop bits\n");
        return (FALSE);
}
/* Set input parity option */
if (parity != 'n')
    options.c_iflag |= INPCK;
tcflush(fd, TCIFLUSH);
options.c_cc[VTIME] = 150; /* Set to more than 15 seconds*/
options.c_cc[VMIN] = 0; /* Update the options and do it NOW */
if (tcsetattr(fd, TCSANOW, &options) != 0)
{
    perror("SetupSerial 3");
    return (FALSE);
}
options.c_lflag &= ~(ICANON | ECHO | ECHOE | ISIG); /*Input*/
options.c_oflag &= ~OPOST; /*Output*/
return (TRUE);
}

int main(int argc, char *argv[])
{
    int fd, c=0, res;
    char *dev;

    char buf[256];

    printf("Start...\n");
    if (argc == 2)
        dev = argv[1];
    else
        dev = UART_DEVICE;
    fd = open(dev, O_RDWR);

    if (fd < 0) {
        perror(UART_DEVICE);
        exit(1);
    }
}

```

```
    printf("Open...\n");
printf("bandrate %d...\n",BAUDRATE);
    set_speed(fd,BAUDRATE);
if (set_Parity(fd,8,1,'N') == FALSE) {
    printf("Set Parity Error\n");
    exit (0);
}

printf("Reading...\n");
while(1) {
    res = read(fd, buf, 255);

    if(res==0)
        continue;
    buf[res]=0;

    printf("%s", buf);

    if (buf[0] == 0x0d)
        printf("\n");
write(fd,buf,res);

    if (buf[0] == '@') break;
}

printf("Close...\n");
close(fd);

return 0;
}
```