Rugged Board

UART

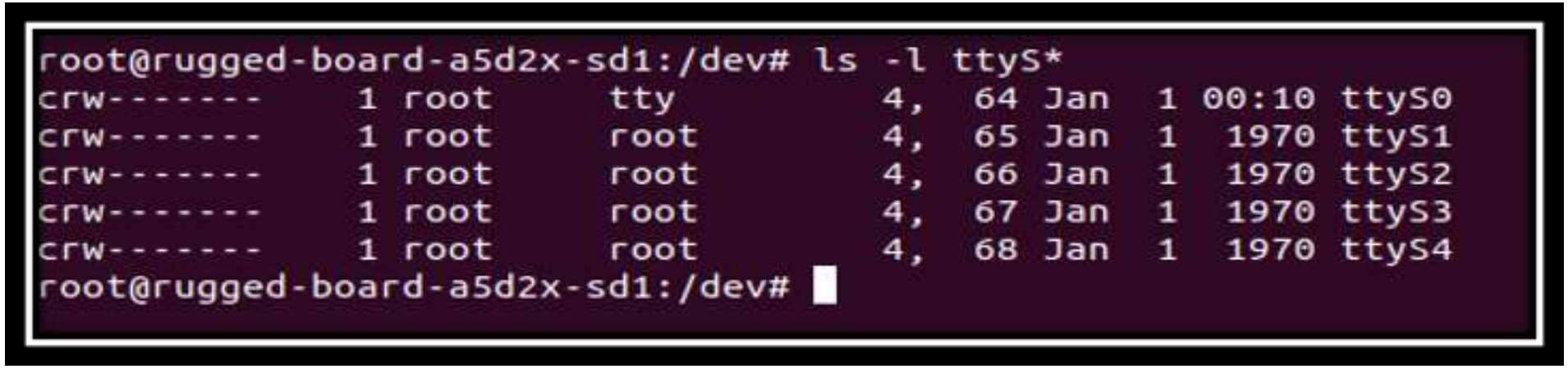
https://community.ruggedboard.com

In the kernel, the UART driver is implemented based on the TTY (TeleTYpewriter)framework.

Many types of terminal drivers are based on TTY, and the UART device is registered as a ttySx character device
('x' stands for the device number).

In RuggedBoard there are Five UARTS

UART1 ---> ttyS0 (USB Debug Port (UART to USB) /TTL Debug Port (3.3V))
UART0 ---> ttyS1 (RS232)
UART4 ---> ttyS4 (RS232)
UART2 ---> ttyS2 (RS485)
UART3 ---> ttyS3 (Mikro Bus)



Terminal Attributes

int tcgetattr(int fd, struct termios* info);

The tcgetattr() copies current settings from tty driver associated to the open file fd into the struct pointed by info. Returns 0 on success and -1 on error

int tcsetattr(int fd,int when,struct termios* info);

The tcsetattr() sets the tty driver associated to the open file fd with the settings given in the struct pointed by info. The when argument tells when to update the driver settings.

when argument can take following values:

TCSANOW: update driver settings immediately

TCSADRAIN: wait until all o/p already queued in the driver has been transmitted to the terminal and then update the driver

TCSAFLUSH: wait for o/p queue to be emptied + flush all queued i/p data and then update the driver

Termios structure

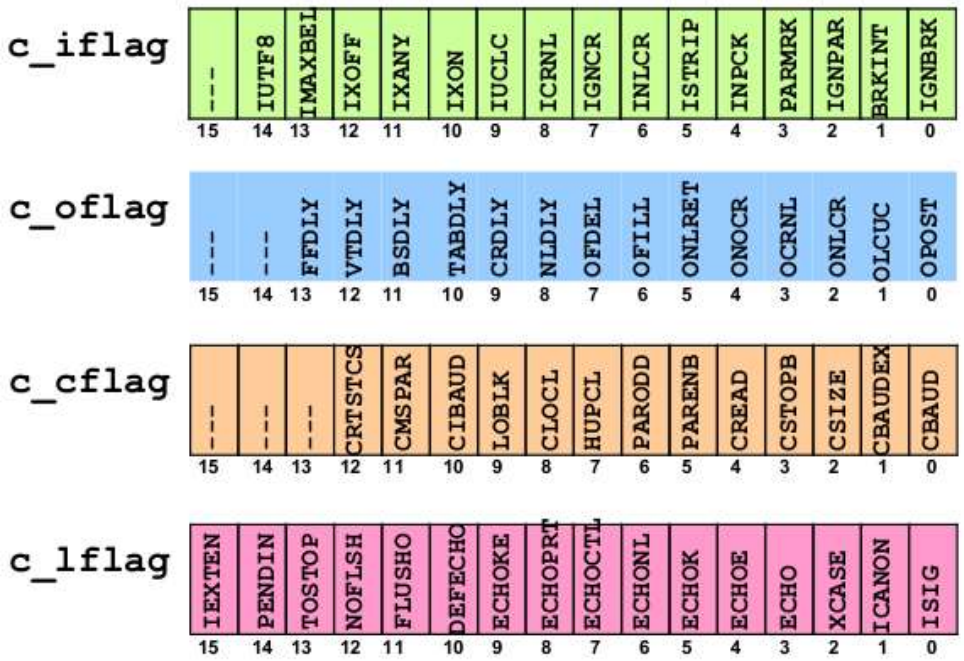
Some important members of the termios structure that of our interest right now are shown below:

```
DESCRIPTION
       The termios functions describe a general  terminal  interface  that  is
       provided to control asynchronous communications ports.

   The termios structure
       Many  of the functions described here have a termios_p argument that is
       a pointer to a termios structure.  This structure contains at least the
       following members:

           tcflag_t c_iflag;      /* input modes */
           tcflag_t c_oflag;      /* output modes */
           tcflag_t c_cflag;      /* control modes */
           tcflag_t c_lflag;      /* local modes */
           cc_t     c_cc[NCCS];   /* special characters */
```

Code snippet to test the echo flag in the c_lflag member of termios structure:

```
struct termios info;
tcgetattr(0, &info);
if((info.c_lflag & ECHO) == 0)
        printf("echo is off, since its bit is 0");
else
        printf("echo is on, since its bit is 1");
```

Individual Bits of termios Flags



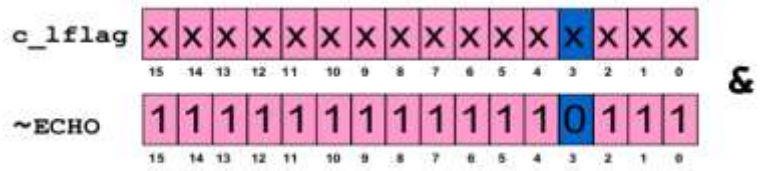
Changing Attributes of Terminal Driver

Three steps to change the attributes of a terminal driver:

- Get the attributes from the driver
- Modify the attribute(s) you need to change
- Send these revised attributes back to the driver

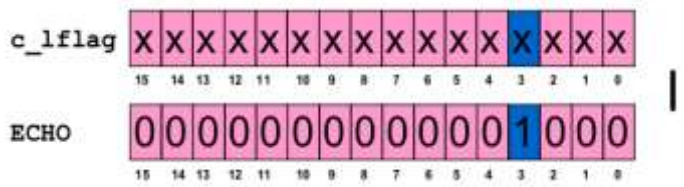
Making the echo Flag OFF

```
struct termios info;
tcgetattr(0, &info);
info.c_lflag = info.c_lflag &  ~ECHO;
tcsetattr(0, TCSANOW, &info);
```



Making the echo Flag ON

```
struct termios info;
tcgetattr(0, &info);
info.c_lflag = info.c_lflag |  ECHO;
tcsetattr(0, TCSANOW, &info);
```



RB-UART

UART3 ---> ttyS3 (Mikro Bus) Loop Back

Step 1: Use open() system call

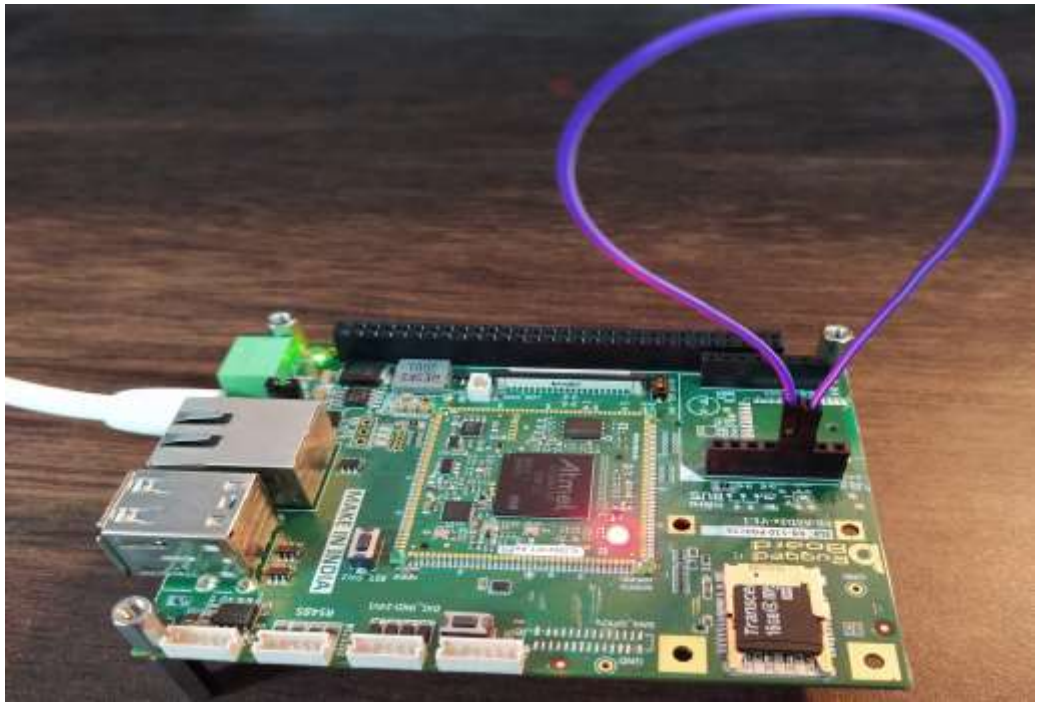
```
fd = open("/dev/ttyS3" O_RDWR | O_NOCTTY | O_SYNC);'
```

Step 2: Set terminal Attributes

```
struct termios tty;
tcgetattr(fd, &tty)
cfsetispeed(&tty, (speed_t)speed);
tty.c_cflag |= (CLOCAL | CREAD);    /* ignore modem controls */
tty.c_cflag &= ~CSIZE;
tty.c_cflag |= CS8;         /* 8-bit characters */
tty.c_cflag &= ~PARENB;     /* no parity bit */
tty.c_cflag &= ~CSTOPB;     /* only need 1 stop bit */
tty.c_cflag &= ~CRTSCTS;    /* no hardware flowcontrol */
tty.c_iflag = IGNPAR;
tty.c_lflag = 0;
tty.c_cc[VMIN] = 1;
tty.c_cc[VTIME] = 1;
tcsetattr(fd, TCSANOW, &tty)
```

Step 3: Use write () and read() system call

```
write(fd,buf,sizeof(buf));
        and
read(fd, buf,sizeof(buf));
```



RB-UART

UART3 ---> ttyS1 or ttyS4 (RS232) Loop Back

Step 1: Use open() system call

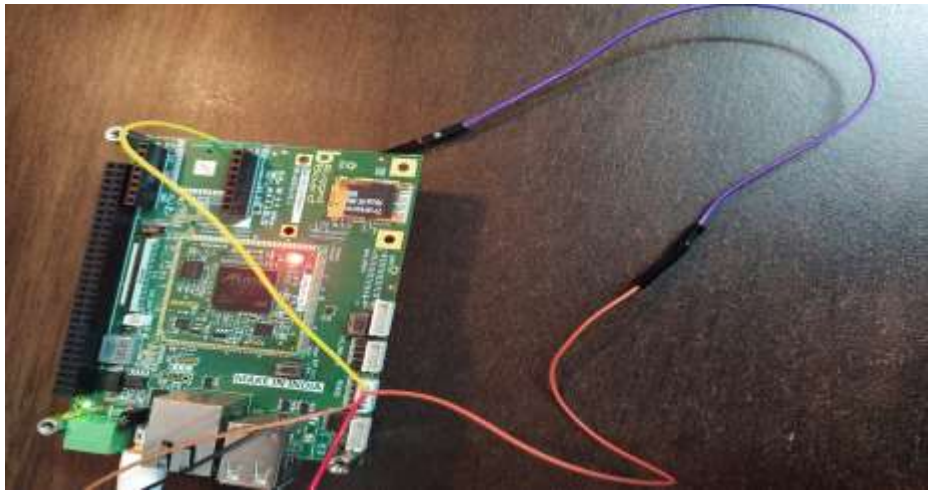
```
fd = open("/dev/ttyS1" O_RDWR | O_NOCTTY | O_SYNC);'
```

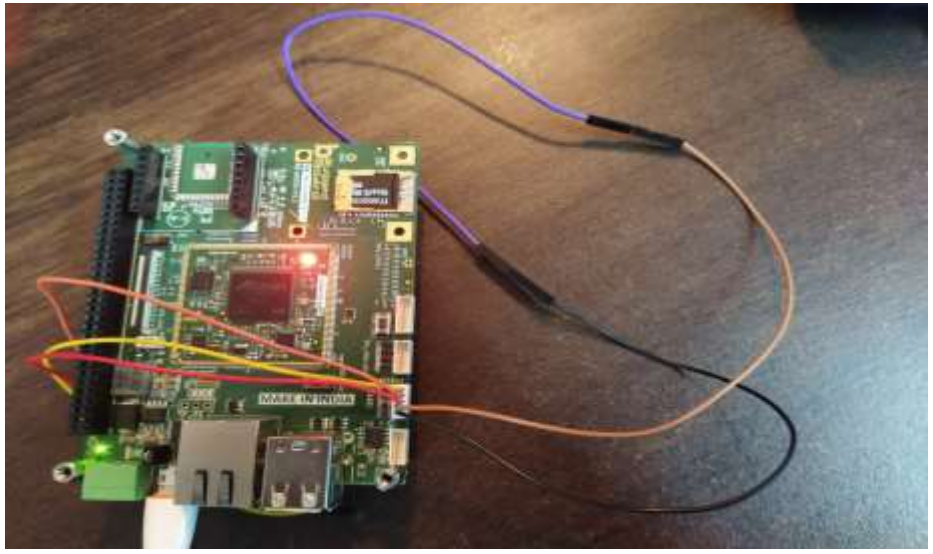
Step 2: Set terminal Attributes

```
struct termios tty;
tcgetattr(fd, &tty)
cfsetispeed(&tty, (speed_t)speed);
tty.c_cflag |= (CLOCAL | CREAD);    /* ignore modem controls */
tty.c_cflag &= ~CSIZE;
tty.c_cflag |= CS8;         /* 8-bit characters */
tty.c_cflag &= ~PARENB;     /* no parity bit */
tty.c_cflag &= ~CSTOPB;     /* only need 1 stop bit */
tty.c_cflag &= ~CRTSCTS;    /* no hardware flowcontrol */
tty.c_iflag = IGNPAR;
tty.c_lflag = 0;
tty.c_cc[VMIN] = 1;
tty.c_cc[VTIME] = 1;
tcsetattr(fd, TCSANOW, &tty)
```

Step 3: Use write () and read() system call

```
write(fd,buf,sizeof(buf));
        and
read(fd, buf,sizeof(buf));
```





Open Discussions

Connect-us

Developer Wiki