```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <fcntl.h>
#include <string.h>

#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>

#include <linux/types.h>
#include <linux/spi/spidev.h>


static int verbose;

static void do_read(int fd, int len)
{
        unsigned char   buf[32], *bp;
        int             status;

        /* read at least 2 bytes, no more than 32 */
        if (len < 2)
                len = 2;
        else if (len > sizeof(buf))
                len = sizeof(buf);
        memset(buf, 0, sizeof buf);

        status = read(fd, buf, len);
        if (status < 0) {
                perror("read");
                return;
        }
        if (status != len) {
                fprintf(stderr, "short read\n");
                return;
        }

        printf("read(%2d, %2d): %02x %02x,", len, status,
                buf[0], buf[1]);
        status -= 2;
        bp = buf + 2;
        while (status-- > 0)
                printf(" %02x", *bp++);
        printf("\n");
}

static void do_msg(int fd, int len)
{
        struct spi_ioc_transfer xfer[2];
        unsigned char           buf[32], *bp;
        int                     status;

        memset(xfer, 0, sizeof xfer);
        memset(buf, 0, sizeof buf);
```

```
        if (len > sizeof buf)
                len = sizeof buf;

        buf[0] = 0xaa;
        xfer[0].tx_buf = (unsigned long)buf;
        xfer[0].len = 1;

        xfer[1].rx_buf = (unsigned long) buf;
        xfer[1].len = len;

        status = ioctl(fd, SPI_IOC_MESSAGE(2), xfer);
        if (status < 0) {
                perror("SPI_IOC_MESSAGE");
                return;
        }

        printf("response(%2d, %2d): ", len, status);
        for (bp = buf; len; len--)
                printf(" %02x", *bp++);
        printf("\n");
}

static void dumpstat(const char *name, int fd)
{
        __u8    mode, lsb, bits;
        __u32   speed;

        if (ioctl(fd, SPI_IOC_RD_MODE, &mode) < 0) {
                perror("SPI rd_mode");
                return;
        }
        if (ioctl(fd, SPI_IOC_RD_LSB_FIRST, &lsb) < 0) {
                perror("SPI rd_lsb_fist");
                return;
        }
        if (ioctl(fd, SPI_IOC_RD_BITS_PER_WORD, &bits) < 0) {
                perror("SPI bits_per_word");
                return;
        }
        if (ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed) < 0) {
                perror("SPI max_speed_hz");
                return;
        }

        printf("%s: spi mode %d, %d bits %sper word, %d Hz max\n",
                name, mode, bits, lsb ? "(lsb first) " : "", speed);
}

int main(int argc, char **argv)
{
        int             c;
        int             readcount = 0;
        int             msglen = 0;
        int             fd;
        const char      *name;
```

```
        while ((c = getopt(argc, argv, "hm:r:v")) != EOF) {
                switch (c) {
                case 'm':
                        msglen = atoi(optarg);
                        if (msglen < 0)
                                goto usage;
                        continue;
                case 'r':
                        readcount = atoi(optarg);
                        if (readcount < 0)
                                goto usage;
                        continue;
                case 'v':
                        verbose++;
                        continue;
                case 'h':
                case '?':
usage:
                        fprintf(stderr,
                                "usage: %s [-h] [-m N] [-r N] /dev/spidevB.D\n",
                                argv[0]);
                        return 1;
                }
        }

        if ((optind + 1) != argc)
                goto usage;
        name = argv[optind];

        fd = open(name, O_RDWR);
        if (fd < 0) {
                perror("open");
                return 1;
        }

        dumpstat(name, fd);

        if (msglen)
                do_msg(fd, msglen);

        if (readcount)
                do_read(fd, readcount);

        close(fd);
        return 0;
}
```