Software

SPI (Master/Slave) Config & Example

There are 4 available SPI controllers available. Each can be configured as a master or slave device.

In this tutorial we will briefly go over configuring the hardware & kernel followed by a C sample demonstrating some master & slave interaction.

Enabling SPI in the Linux Kernel

```
CONFIG_SPI=y
CONFIG_SPI_MASTER=y
CONFIG_SPI_SUNXI=y
CONFIG_SUNXI_SPI_NDMA=y
CONFIG_SPI_SPIDEV=y
CONFIG_SPI_SLAVE=y
```

Sample sys_config (script.bin)

```
[spi_board0]
modalias = "spidev"
max_speed_hz = 8000000
bus_num = 0
chip_select = 0
mode = 1
full_duplex = 0
manual_cs = 0
is_slave = 1

[spi_board1]
modalias = "spidev"
max_speed_hz = 8000000
bus_num = 1
chip_select = 0
mode = 1
full_duplex = 0
manual_cs = 0

[spi_board2]
modalias = "spidev"
max_speed_hz = 8000000
bus_num = 2
chip_select = 0
mode = 1
full_duplex = 0
manual_cs = 0

[spi_devices]
```

```
spi_dev_num = 3

[spi0_para]
spi_used = 1
spi_cs_bitmap = 1
spi_cs0 = port:PI10<2><default><default><default>
spi_cs1 =
spi_sclk = port:PI11<2><default><default><default>
spi_mosi = port:PI12<2><default><default><default>
spi_miso = port:PI13<2><default><default><default>

[spi1_para]
spi_used = 1
spi_cs_bitmap = 1
spi_cs0 = port:PI16<2><default><default><default>
spi_cs1 =
spi_sclk = port:PI17<2><default><default><default>
spi_mosi = port:PI18<2><default><default><default>
spi_miso = port:PI19<2><default><default><default>

[spi2_para]
spi_used = 1
spi_cs_bitmap = 1
spi_cs0 = port:PB14<2><default><default><default>
spi_cs1 =
spi_sclk = port:PB15<2><default><default><default>
spi_mosi = port:PB16<2><default><default><default>
spi_miso = port:PB17<2><default><default><default>

[spi3_para]
spi_used = 0
spi_cs_bitmap = 1
spi_cs0 = port:PA05<3><default><default><default>
spi_cs1 = port:PA09<3><default><default><default>
spi_sclk = port:PA06<3><default><default><default>
spi_mosi = port:PA07<3><default><default><default>
spi_miso = port:PA08<3><default><default><default>
```

Note: An equivelnt sample device tree source (DTS) for use with newer versions of the kernel will be made available soon.

Check your dmesg to make sure the devices are properly configured.

```
root@awsom:~# dmesg | grep spi
[    1.024251] [spi-inf] Found 3 spi devices in config files
[    1.031558] [spi-inf] boards num modalias        max_spd_hz       bus_num  cs   mode slave
```

```
[    1.042494] [spi-inf] 0              spidev          8000000         0       0       0x1     Y
[    1.053388] [spi-inf] 1              spidev          8000000         1       0       0x1     N
[    1.064281] [spi-inf] 2              spidev          8000000         2       0       0x1     N
[    1.069217] [spi-inf] sunxi_spi_probe: spi0 dma type: normal
[    1.077790] [spi-inf] sunxi_spi_set_mclk: spi0 source = sdram_pll_p, src_clk = 408000000, mclk 81600000
[    1.084932] [spi-inf] allwinners SoC SPI Driver loaded for Bus SPI-0 as slave device
[    1.089598] [spi-inf] sunxi_spi_probe: spi1 dma type: normal
[    1.093173] [spi-inf] bus num = 1, spi used = 1
[    1.097611] [spi-inf] sunxi_spi_probe: spi1 cs bitmap: 0x1
[    1.106160] [spi-inf] sunxi_spi_set_mclk: spi1 source = sdram_pll_p, src_clk = 408000000, mclk 81600000
[    1.112032] sun7i-spi sun7i-spi.1: master is unqueued, this is deprecated
[    1.121001] [spi-inf] sunxi_spi_probe: reuuimlla's SoC SPI Driver loaded for Bus SPI1 with 2 Slaves at most
[    1.130410] [spi-inf] sunxi_spi_probe: spi1 driver probe succeed, base f00d6000, irq 43, dma_id_rx 25, dma_id_tx 25
[    1.135058] [spi-inf] sunxi_spi_probe: spi2 dma type: normal
[    1.138645] [spi-inf] bus num = 2, spi used = 1
[    1.143084] [spi-inf] sunxi_spi_probe: spi2 cs bitmap: 0x1
[    1.151593] [spi-inf] sunxi_spi_set_mclk: spi2 source = sdram_pll_p, src_clk = 408000000, mclk 81600000
[    1.157454] sun7i-spi sun7i-spi.2: master is unqueued, this is deprecated
[    1.166402] [spi-inf] sunxi_spi_probe: reuuimlla's SoC SPI Driver loaded for Bus SPI2 with 2 Slaves at most
[    1.175808] [spi-inf] sunxi_spi_probe: spi2 driver probe succeed, base f00d8000, irq 44, dma_id_rx 26, dma_id_tx 26
```

Sample C program demonstrating Master & Slave

compile with -DSLAVE to compile as slave. You will also need to include spidev.h or equivalent prototypes in your build path.

```
gcc spi.c -o master
gcc spi.c -DSLAVE -o slave



./master <dev> <speed> <len> <udelay>
./slave <dev> <speed> <len>



#include <stdint.h>
#include <unistd.h>
#include <sys/time.h>
#include <signal.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <getopt.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/types.h>
//#include <linux/spi/spidev.h>
#include "spidev.h"

//#define DEBUG

char spi_dev[16];
long speed=50000;
long udelay=0;
struct spi_ioc_transfer xfer[2];
struct timeval start, finish;
long msec;
int file;

unsigned short crc16(unsigned char* data_p, unsigned char length){
    unsigned char x;
    unsigned short crc = 0xFFFF;

    while (length--){
        x = crc >> 8 ^ *data_p++;
        x ^= x>>4;
        crc = (crc << 8) ^ ((unsigned short)(x << 12)) ^
             ((unsigned short)(x <<5)) ^ ((unsigned short)x);
    }
    return crc;
}
long timevaldiff(struct timeval *starttime, struct timeval *finishtime
)
{
    long msec;
    msec=(finishtime->tv_sec-starttime->tv_sec)*1000;
    msec+=(finishtime->tv_usec-starttime->tv_usec)/1000;
    return msec;
}
int spi_init(char filename[40])
{
    int file;
    __u8    mode, lsb, bits;
    __u32 speed=speed;

    if ((file = open(filename,O_RDWR)) < 0)
    {
        printf("Failed to open the bus.");
```

```
        /* ERROR HANDLING; you can check errno to see what went wrong
*/
        exit(1);
    }

    /*multiple modes:
      mode |= SPI_LOOP;
      mode |= SPI_CPHA;
      mode |= SPI_CPOL;
      mode |= SPI_LSB_FIRST;
      mode |= SPI_CS_HIGH;
      mode |= SPI_3WIRE;
      mode |= SPI_NO_CS;
      mode |= SPI_READY;
      if (ioctl(file, SPI_IOC_WR_MODE, &mode)<0)       {
      perror("can't set spi mode");
      return;
      }
     */

    if (ioctl(file, SPI_IOC_RD_MODE, &mode) < 0)
    {
        perror("SPI rd_mode");
        return;
    }
    if (ioctl(file, SPI_IOC_RD_LSB_FIRST, &lsb) < 0)
    {
        perror("SPI rd_lsb_fist");
        return;
    }
    if (ioctl(file, SPI_IOC_RD_BITS_PER_WORD, &bits) < 0)
    {
        perror("SPI bits_per_word");
        return;
    }
    if (ioctl(file, 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",
        filename, mode, bits, lsb ? "(lsb first) " : "", speed);

    xfer[0].cs_change = 0; // Keep CS activated
    xfer[0].delay_usecs = udelay;
    xfer[0].speed_hz = speed;
    xfer[0].bits_per_word = 8; // 8-bites per word only
```

```
    xfer[1].cs_change = 0;
    xfer[1].delay_usecs = 0;
    xfer[1].speed_hz = speed;
    xfer[1].bits_per_word = 8;

    return file;
}

int spi_read(int add1,int add2,int nbytes,char* buffer,int file)
{
    static char cmd[3];
    int status;

    memset(buffer, 0, nbytes);
    cmd[0] = 0x01;
    cmd[1] = add1;
    cmd[2] = add2;
    xfer[0].tx_buf = (unsigned long)cmd;
    xfer[0].len = 3; /* Length of  command to write*/

    xfer[1].rx_buf = (unsigned long) buffer;
    xfer[1].len = nbytes; /* Length of Data to read */

    status = ioctl(file, SPI_IOC_MESSAGE(2), xfer);
    if (status < 0)
    {
        perror("SPI_IOC_MESSAGE");
    }
    return status;
}
int spi_slave_read(int nbytes,char* buffer, int file)
{
    int status;

    memset(buffer, 0, nbytes);
    xfer[0].rx_buf = (unsigned long) buffer;
    xfer[0].len = nbytes; /* Length of Data to read */
    status = ioctl(file, SPI_IOC_MESSAGE(1), xfer);
    if (status < 0)
    {
        perror("SPI_IOC_MESSAGE");
    }
    return status;
}

void spi_write(int nbytes,char* buffer,int file)
{
    int status;
```

```
    xfer[0].tx_buf = (unsigned long)buffer;
    xfer[0].len = nbytes; /* Length of command to write */
    status = ioctl(file, SPI_IOC_MESSAGE(1), xfer);
    if (status < 0)
    {
        perror("SPI_IOC_MESSAGE");
        return;
    }
}

void dump(char *buffer, int len){
    int x;
    for(x=0; x<len; x++){
        printf("0x%X ",*(buffer+x));
    }
    printf("\n");
}

void quit(int sig)
{
    printf("Closing...\n");
    close(file);
    exit(0);
}
main ( int argc, char **argv ) {
    int x, y, len,status;
    long frame, last_frame, good, bad;
    unsigned short crc;
    char *rd,*wr;

    signal (SIGINT, quit);
    if(argc!=4 && argc!=5){
#ifdef SLAVE
        printf("Usage: ./slave <dev> <speed> <len>\n\t" \
            "./slave 2 5000000 64\n");
#else
        printf("Usage: ./master <dev> <speed> <len> <udelay>\n\t" \
            "./master 1 5000000 64 100\n");
#endif
        exit(0);
    }

    good=bad=0;
    frame=1;
    sprintf(spi_dev,"/dev/spidev%s.0",argv[1]);
    speed=atoi(argv[2]);
    len= atoi(argv[3]);
    if(argc==5)
        udelay=atoi(argv[4]);
```

```
#ifdef SLAVE /* Run as Slave */
    rd=(char*)malloc(3);
    wr=(char*)malloc(len);
    printf("Slave: %s speed=%s len=%s\n", spi_dev, argv[2], argv[3]);

    file=spi_init(spi_dev);
    while(1){
        status=spi_slave_read(3,rd,file);

        if(*rd==0x01){
            frame=(*(rd+1)+(*(rd+2)<<8));
            *(wr)=frame&0xff;
            *(wr+1)=(frame>>8)&0xff;
            *(wr+2)=3;
            *(wr+3)=4;
            for(y=4; y<len-2; y++)
                *(wr+y)=(frame%0xff);
            crc=crc16(wr,len-2);
            *(wr+(len-1))=crc&0xff;
            *(wr+(len-2))=(crc>>8)&0xff;
#ifdef DEBUG
            dump(wr,10);
            printf("\n");
#endif
            spi_write(len,wr,file);
            if(last_frame==frame)
                printf("Slave resend frame %ld %d bytes \n", frame, le
n);
        }
        last_frame=frame;
    }
#else /* Run as Master */
    wr=(char*)malloc(3);
    rd=(char*)malloc(len);
    printf("Master: %s speed=%s len=%s\n", spi_dev, argv[2], argv[3]);
    x=0;
    file=spi_init(spi_dev);
    gettimeofday(&start, NULL);
    while(1){
        //gettimeofday(&start, NULL);
        spi_read(frame&0xff, ((frame>>8)&0xff), len, rd, file);
        if(*(rd+2)==3 && *(rd+3)==4){
            crc=crc16(rd,len);
            if(crc==0){
                good++;
                frame++;
            }else{
                bad++;
```

```
                printf("invalid crc %X for frame %ld. retry!\n", crc,
frame);
            }
#ifdef DEBUG
            printf("Master Read ");
            dump(rd,10);
#endif
            gettimeofday(&finish, NULL);
            msec = timevaldiff(&start, &finish);
            if(frame%1000==0) /* show update every 1K frames */
                printf("Master read %ld kb in %ldms %ld/%ld\n",
                    (len*frame)/1000, msec, good, bad);
        }else{
            bad++;
#ifdef DEBUG
            printf("Invalid data!\n");
            dump(rd,10);
#endif
        }

    }
#endif

    return 0;
}
```

Unique solution ID: #1026
Author: Michael McAndrew
Last update: 2015-05-19 08:20