ECE 471 – Embedded Systems
Lecture 16

Vince Weaver

http://www.eece.maine.edu/∼vweaver

vincent.weaver@maine.edu

30 October 2014

Announcements

- HW7 will be posted tonight

HW6 Results

- Delayed due to lack of people in class

SPI bus on Linux

- `modprobe spidev`
- `modprobe spi-bcm2708`
- `dmesg | grep spi`

SPI dev interface

- `https://www.kernel.org/doc/Documentation/spi/spidev`
- `/dev/spidevB.C` (B=bus, C=slave number).
  On pi it is `/dev/spidev0.0`
- Other useful info in `/sys/devices/.../spiB.C`, `/sys/class/spidev/spidevB.C`
- To open the device, do something like the following
  `spi_fd=open("/dev/spidev0.0",O_RDWR);`

- To set the write mode, use ioctl:

  ```
  int mode=SPI_MODE_0;
  result = ioctl(spi_fd, SPI_IOC_WR_MODE, &mode);
  ```

  Modes can be SPI_MODE_0 through 3, or else you can build them out of SPI_CPOL and SPI_CPHA values. Current mode can be read back with SPI_IOC_RD_MODE

- To set the bit order, use ioctl:

  ```
  int lsb_mode=0;
  result = ioctl(spi_fd, SPI_IOC_WR_LSB_FIRST, &lsb_mode);
  ```

  Current can be read with SPI_IOC_RD_LSB_FIRST
  Get/Set if MSB is first (common) or LSB is first.
  Empty bits padded to left with zeros no matter what the

setting.

- ioctl SPI_IOC_RD_BITS_PER_WORD, SPI_IOC_WR_BITS
  Number of bits in each transfer word. Default (0) is 8 bits.

- ioctl SPI_IOC_RD_MAX_SPEED_HZ, SPI_IOC_WR_MAX_
  Set the maximum clock speed.

- By default using `read()` or `write()` on the device node will only do half-duplex.

- For full duplex support you need something like the

following:

```c
#define LENGTH 3
int result;
struct spi_ioc_transfer spi;
unsigned char data_out[LENGTH]={0x1,0x2,0x3};
unsigned char data_in[LENGTH];

/* Setup full-duplex transfer of 3 bytes */
spi.tx_buf    = (unsigned long)&data_out;
spi.rx_buf    = (unsigned long)&data_in;
spi.len       = LENGTH;
spi.delay_usecs      = 0 ;
spi.speed_hz         = 100000 ;
spi.bits_per_word    = 8 ;
spi.cs_change        = 0 ;

/* Run one full-duplex transaction */
result = ioctl(spi_fd, SPI_IOC_MESSAGE(1)), &spi) ;
```

Analog Digital Converters on Raspberry PI

- Unlike many other embedded boards, the Pi has no A/D converters built in.
- You're stuck using SPI or i2c devices

MCP3008

- For HW#7 we'll use the MCP3008 8-port 12-bit SPI A/D converter
- up to 100ksp (samples per second)
- Returns 10-bits of accuracy
- 8 single-ended inputs (vs ground) or 4 "pseudo-differential" inputs (vs each other)
- Config sent in each request packet

- Clock frequency must be long enough that the A/D has time to convert
- $V_{IN} = \frac{value \times V_{REF}}{1024}$
- Send a 1 as a start bit
- Send a SGL (1) /DIFF (0) bit if single or differential mode
- Send 3 bits indicating channel
- Wait 1 more cycle

- Will respond with 0, then 10 bits of sample, then 0s forever until stop clocking

MCP3008 μcontroller mode

- Datasheet describes way to easily use from a device
- Send 3 bytes. First has value '1' (the start bit). The second has the top 4 bits being single/diff followed by 3 bits of which channel you want. The rest is all 0s for padding.
- You read back 3 bytes. First 13 bits are don't care (ignore) followed by 0 then the 10 bits of sample.
- XXXXXXXX XXXXX098 76543210

TMP36

- Linear temperature sensor
- The temperature can be determined with the following equation:
  $deg\_C = (100 \times voltage) - 50$
- Also the following might be useful:
  $deg\_F = (deg\_C \times \frac{9}{5}) + 32$
- Be careful hooking up! If vdd/gnd switched it heats up to scalding temperatures (the datasheet lists the pinout

from the bottom). If you catch it in time doesn't seem to be permanently damaged.

Floating Point in C

- Converting int to floating point:

```
int value=45;
double temp;

temp=value;            // works
temp=(float)value;     // casts make the conversion explicit
                       // but can potentially hide bugs
```

- float vs double
  float is 32-bit, double 64-bit

- Constants 9/5 vs 9.0/5.0

The first is an integer so just “1”. Second is expected 1.8.

- Printing. First prints a double. Second prints a double with only 2 digits after decimal.

```
printf("%lf\n",temp);
printf("%.2lf\n",temp);
```