ECE 471 – Embedded Systems
Lecture 13

Vince Weaver

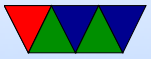
http://www.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

17 October 2013

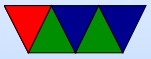
Announcements

- Homework #3 will be posted today.
- Drop off your board to be soldered if you haven't already.
- Midterm coming up on Tuesday, October 22nd.

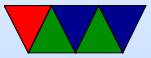


Midterm Review

- Closed notes
- Everything from HW1, HW2, lectures
- Know what makes an embedded system
- Advantages/Disadvantages of Operating System
- Assembly language: should be able to explain short chunks of code, write short programs. Won't expect you to know obscure instructions.

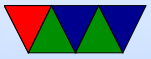


- Roughly 1/2 short answer, 1/2 assembly language



System Busses

- Older busses often exposed CPU pins directly to connector: Apple II, S-100, ISA
- This was not sustainable, if only because number of CPU pins grew rapidly. Also speed issues.

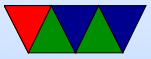


Parallel vs Serial Busses

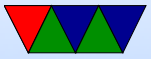
- Originally most busses were Parallel. More bits at a time means higher bandwidth. IDE, Parallel Port, 32-bit PCI, 64-bit PCI

- Problems with parallel: keeping signals in sync. As busses go faster, skew comes into things. Wire length matters. Power issues with driving wide busses.

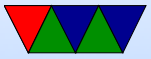
- Newer busses are serial: SATA, PCIe, USB, Firewire, etc. Also advantage of having fewer wires to route.



- People (especially HPC) still grumble about speed of PCIe

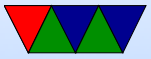


Embedded Busses

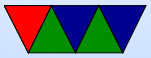


GPIOs

- General Purpose Input/Output
- Programmable pins on microcontroller
- Can be used as input or output
- Can be hooked up to an interrupt
- Can be tied together and treated as a port
- Can be hooked to an LED



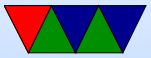
- Have different input/output voltage tolerances



GPIO – Interface

- Kernel can control GPIOs. Driver can hook together to make other busses (i2c) or do simple tasks (such as control an LED)

- Can also drive directly from userspace.
  - Export for user control
    ```
    echo 13 > /sys/class/gpio/export
    ```
  - Change direction
    ```
    echo "out" > /sys/class/gpio/gpio13/direction
    ```



```
echo "in" > /sys/class/gpio/gpio13/direction
```

- Change value:

```
echo 1 > /sys/class/gpio/gpio13/value
```

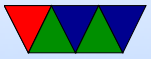
- Read value:

```
cat /sys/class/gpio/gpio13/value
```

- Unexport:

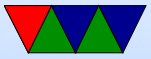
```
echo 13 > /sys/class/gpio/unexport
```

- Cannot access a gpio a kernel driver is using
- Can use C (or other languages) to access the GPIOs. If writing, remember to re-wind (or open-close) before each write.

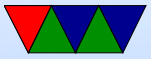


Gumstix GPIOs

- GPIO multiplexing setup by the bootloader (uboot)
  You can modify this on running system but tricky
- 1.8V
- Many of GPIOs already in use (for i2c, etc). Numbers indicated on the 40-pin header description
- GPIOs you can use:
  - Pin 4 - gpio114 – pendown from touch screen (if in

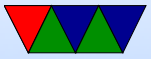


use), input only
- Pin 19 - gpio170 – 1-wire interface (if in use), output only
- Pin 27 - gpio146, Pin 29 - gpio147 – input or output
- Pin 28 - gpio145, Pin 30 - gpio144 – LCD (if in use). output only



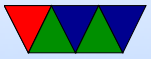
i2c

- Inter-Integrated Circuit, Invented by Philips (now NXP) in 1982
- Two-wires
- Since 2006, no licensing fees (though do have to pay to reserve number)



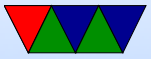
Why is i2c popular?

- Stable standard
- Relatively easy to implement
- Not many wires
- Good enough
- Cheap



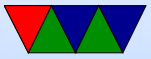
Uses of i2c

- SMbus
- DDC (video card / monitor communication)
- Configuring SDRAM
- Temp sensor and fan chips on motherboards
- Wii nunchuck

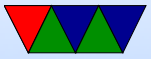


Protocol Overview

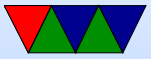
- Serial Data Line (SDA) and Serial Clock (SCL), Open Drain, Pulled up by resistors
- Open drain means output can be wired together
- 7-bit (or 10-bit) address speeds
- Standard=100kbits/s, slow=10kbits/s, fast=400kbits/s fast plus 1Mbits/s, high 3.4Mbits/s (actual transfers slower due to overhead)



- Length of bus limited to a few meters
- Master (generates clock, init transaction), Slaves (responds)
- Can be multiple masters and slaves
- Master sends start bit, 7-bit address of slave, then read/write bit
- Slave responds with ACK then interacts
- Address and Data set Most-significant Bit first



- Start bit is SDA high-low while SCL high. Stop is SDA low-high while SCL high.
- To transmit bit, set SCL high, waits 4us, sets low
- Clock stretching: slave can hold SCL low until it is done processing
- Arbitration: masters monitor SDA and won't start unless idle. Deterministic arbitration.



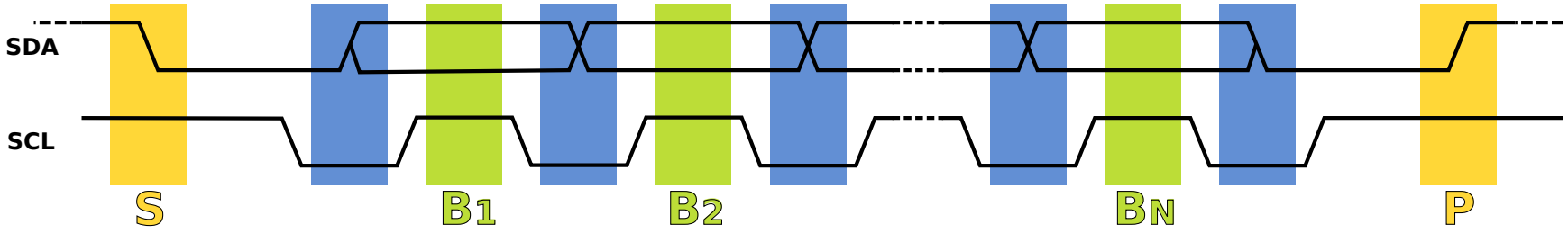
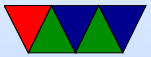
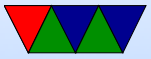


Figure 1: Protocol diagram from Wikipedia



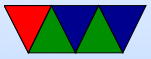
i2c and Linux

- Like with GPIOs, kernel can drive it, or be exposed to userspace
- i2c-dev module must be installed (and i2c driver)
- Open the device node, /dev/i2c-3
- Use ioctls I2C_SLAVE to set the address of the device we wish to talk to.
- Use standard read or write calls to communicate with



the device

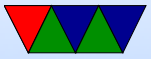
- Close the device when done.
- i2c slave addresses are 7 bits, but when sent the r/w bit is put at end. This can be confusing; some spec sheets will list a slave address as 0x38 (7 bits) but Linux exports this as 0x70 (0x38 shifted left by 1).



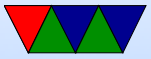
Sample i2c Linux code

For more details on this, see the HW3 handout.

```
unsigned char buffer[17];
int display_fd;

/* open */
display_fd = open("/dev/i2c-3", O_RDWR);
if (display_fd < 0) fprintf(stderr,"Error!\n");

/* set slave address */
result=ioctl(display_fd, I2C_SLAVE, 0x70);
if (result < 0) fprintf(stderr,"Error!\n");

/* writing */
buffer[0]= HT16K33_REGISTER_SYSTEM_SETUP | 0x01;
```

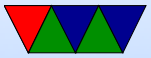


```
if ( (write(display_fd, buffer, 1)) !=1) fprintf(stderr,"Error!\n")

/* closing */
close(display_fd);
```



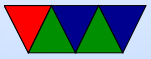
i2c on the Overo

- 1.8V
- default speed is 400kHz
- i2c-1 connects to the power-control circuitry
- i2c-3 connects to the header



i2c on the Overo – detecting

```
i2cdetect -y -r 3
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 -- -- -- -- -- -- --
```



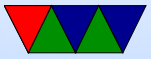
LED Driver Chip

- This is a ht16k33, datasheet available: `http://www.adafruit.com/datasheets/ht16K33v110.pdf`

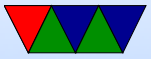
- Supports up to 16x8 LEDs, as well as keypad input. Can dim display, also blink. Common cathode.

  ```
  -|>|- common
  ```

- Works by rapidly scanning all segments fast enough cannot see.



- To set up, write byte commands, high 4 bits command lower 4 bits data.
- To set up full display, write the pointer offset of internal framebuffer, than 16 bytes of on/off data.
- Actual LED hooked up is a BL-Q56D-43UG 4x7 segment Ultra-Green display common cathode.



Benefit of OS

- Code is portable across all machines with i2c bus
- Can use same code on Gumstix, Rasp-Pi, Beaglebone, etc.
- Will probably need to change the bus number (I think it's i2c-1 on pi).