ECE 471 – Embedded Systems Lecture 15

Vince Weaver

`http://web.eece.maine.edu/~vweaver`

`vincent.weaver@maine.edu`

25 October 2016

Announcements

- Don't wait until the last minute for HW#7

HW6 Results

- Make sure you write ASCII '0' to GPIO to pull it low, not 0
  Also you need to point the write to something pointing to '0'. Can't just have write(fd,0,1);
  I was confused how the code worked despite never pulling SDA low, until I realized the sample code was printing all 1s to display.
  If you were checking errors, it would have reported EINVAL if you wrote non-ASCII 0 to the GPIO value.

- Why is it slow? What's the minimum time you can delay when using usleep()? Bitbanging in general?
- static in front of a function or global var means it only is called in this .c file.
  This is slightly different than the meaning of using static in front of a local variable (inside of a function).
- What's missing vs full protocol?
  Didn't implement clock stretching: When stop bit, when releasing SCL to go high, if it stays low the slave isn't ready to move on, should wait until it goes high.

Arbitration
Read byte
Parameterized address

- Error checking. Be aware of errors.

- interrupt sources
  doorbell? more complex on pi2, pi3. IPI is inter-processor

- yes
  for answers a large list of questions in the affirmative?
  stress test? maybe

hold down your finger forever on fsck. Hope last one isn't something like "erase filesystem"

HW7 Notes

- Why is the kernel erroring out if the empty “pad” bit not zero?
  Forward compatibility. You want to make sure that any empty bits stay that way. If you want to add new functionality in the future you have to ensure reserved bits are all zero, otherwise old programs will do unexpected things (or break) if they had been accidentally setting those bits.
  So why were the pad bits non-zero? Bad luck. Local

struct allocated on the stack, so if there were old values on the stack the pad value could be non-zero.

- ADC max is 1023

Project Preview

- Can work in groups
- Embedded system (any type, not just Pi)
- Written in any language (asm, C, python, C++, Java, etc.)
- Do some manner of input and some manner of output using the various capabilities we discussed
- I have a large amount of i2c, spi, and other devices that

you can borrow if you want to try anything interesting.

- Past projects: games, robots, weather stations, motor controllers, music visualization, etc.

- Will be a final writeup, and then a 10 minute presentation and demo in front of the class during last week of classes.

Analog Digital Converters

Floating Point in C

- Converting int to floating point:

```
int value=45;
double temp;

temp=value;              // works
temp=(float)value;       // casts
                         // but ca
```

- 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);
```

Determining worst case behavior.

- Hard on modern processors. Easier on stm32l than on raspberry pi running Linux
- STM32L is in-order. Program in assembly. Turn off interrupts. You know exactly when everything is happening.
- Pi, with OS. Can you disable interrupts?
- Hybrid? Full OS, but with deterministic processors tacked on? BBB PRUs

Common OS strategies

- Event driven – have priorities, highest priority pre-empts lower
- Time sharing – only switch at regular clock time, round-robin

Scheduler example

- Static: Rate Monotonic Scheduling – shortest job goes first
- Dynamic: Earliest deadline first
- Three tasks come in. a. deadline 10s, runs 4s. b. deadline 3s, runs 2s, c. deadline in 5s, runs 1s
- In order they arrive, aaaabbc bad for everyone
- RMS: cbbbaaaa works

- EDF: bbbcaaaa also works.
- Lots of information on various scheduling algorithms

Priority Inversion Example

- Task priority 3 takes lock on some piece of hardware
- Task 2 fires up and pre-empts task 3
- Task 1 fires up and pre-empts task 1, but it needs same HW as task 3. Waits for it. It will never get free.
- Space probes have had issues due to this.

Real Time OS

Who uses realtime?

- Timing critical situations. Cars, medical equipment, space probes, etc.
- Industrial automation. SCADA. Stuxnet.
- Musicians, important to have low-latency when recording
- High-speed trading

PREEMPT Kernel

- Linux PREEMPT_RT
- Faster response times
- Remove all unbounded latencies
- Change locks and interrupt threads to be pre-emptible

Linux PREEMPT Kernel

- What latencies can you get? 10-30us on some x86 machines
- Depends on firmware; SMI interrupts (secret system mode, can't be blocked, emulate USB and like)' Slow hardware; CPU frequency scaling; nohz
- Special patches, recompile kernel
- mlockall() memory in, start threads and touch at beginning, avoid all causes of pagefaults.

Co-operative real-time Linux

- Xenomai
- Linux run as side process, sort of like hypervisor

Other RTOSes

- Vxworks
- Neutrino
- Free RTOS
- Windows CE