ECE 271 – Microcomputer Architecture and Applications Lecture 24

Vince Weaver

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

vincent.weaver@maine.edu

19 April 2022

Announcements

- Don't forget course reviews
- Final exam, more info on that later
- Last Lab this week
- You can keep your lab supplies. Boards might be re-used in DSP class.

Comments on Lab#11 DAC

- To play sound we are interrupting 44.1k times a second
- Is that very efficient?
- At 16MHz, it means being interrupted every 362 cycles depending how long your irq handler is that doesn't leave much time to get anything else done
- Wouldn't it be great if we could pre-calculate say 1000 samples we'd want to output, and then tell the hardware to play them one after another w/o us having to do anything in software?

Direct Memory Access (DMA)

- Read Chapter 19
- Transfer data without the CPU being involved
  - Why not use the CPU?
  - It's a bit slow. Load/store for every byte, CPU busy
- Transfer data between peripherals and memory or memory to peripherals
  - For slow peripherals, CPU doesn't have to wait around
  - For fast peripherals, can improve throughput
  - For high-speed, can reduce interrupt load

DMA Examples

- DAC – can queue up samples to play, and on timer interrupt the value loaded direct from MEM to DAC w/o the CPU involved
- ADC – can read in sampled values to memory and CPU only has to deal with it once enough have built up

AMBA – Advanced Microcontroller Bus Architecture

- ARM standard that devices can connect to (royalty-free)
  - AHB (Advanced high-perf bus) – used on Cortex M
    GPIO (AHB2ENR_GPIOAEN), DAC
  - APB (Advanced Peripheral Bus) – for low-speed devices
    such as LCD (APB1ENR1_LCDEN), DAC
  - ASB (Advanced System Bus) – high speed bus

DMA Controller

- On AHB bus
- Takes orders from CPU, can have data transfers on AHB and APB via the AHB/APB bridge
- Flow-through DMA – data is read from the source, buffered, and written to destination (can be used when src/dst different sizes)

  also useful in memory to memory if can't read/write same cycle
- Fly-by DMA – data directly transferred from src to dst

w/o being read into DMA controller

With/Without DMA

- Without DMA
  - Could busy wait (poll) until device is ready
  - Can also use interrupt – why might that be bad?
    High interrupt loads keep CPU from getting other work done, overhead of running handler each time
  - CPU loads value to register, stores out to device
- With DMA
  - DMA controller notified when device ready
  - Copies data in background

◦ Can optionally send an IRQ to CPU to let it know something happened

Programming DMA – STM32L4

- Cortex M has two DMA controllers, each 7 channels
- Channels in DMA controller hardcoded (sort of like GPIO pin assignments), have to select which one active
- There's a software and hardware priority for which takes precedence

Programming DMA – Registers

- CMAR – channel memory address register
  Address of memory
- CPAR – channel peripheral address register
  Address of device
- CNDTR – channel number of data register
  How much data to transfer
- CCR – channel configuration register
  direction, increment, circular, priority, interrupts

Programming DMA – Circular Buffer

- Can not increment (for example, if copying from register) or auto-increment (if copying from memory)
- Can have circular buffer where it wraps at end
  Why? You can set it to go forever but refill once it has gone past

Programming DMA – Interrupts

- Half-transfer flag HT1F (half the data has been sent)
- Transfer complete (TCIF)
- Error (TEIF) if access memory it shouldn't
- General (GIF) if any of above triggered
- Clear these by writing 1 to the IFCR register

Lab#11 – Something Cool / Making Music

- Only about 5 extra lines plus some lookup tables
- See the textbook

Musical Notes

- Musical notes, A4=440Hz. A4 is pitch#69
- $f = 440 \times 2^{(p-69)/12}$
- Octave has 8 notes, but really 12 notes if you could sharps/flats

Note Lengths

- Have a countdown timer that is set for the length of the note and then counts down until it is done, then picks the next note.
- If 120BPM (bits per minute), then a standard note changes after 1/2 a 44100 cycle so count down from 22050*length.
- It will still sound electronic. To get instrument-like sounds you'd need to mess with the "envelope" (attack, sustain, decay, release)

- You can play multiple channels if you add different frequencies together, just make sure you divide so the value doesn't overflow 4096 (12-bits) or it will wrap around and sound weird.
- Rests / pauses between notes – just output 0 (2048?)
- Switching notes – glitches can happen mid-frequency change if there's a discontinuity. Best way to avoid is only switch frequency at a zero (x-axis) crossing

Advanced music – One pin GPIO

- This is all older machines had
- Can do a square wave of certain frequency. Hard on amplifiers + speakers (lots of higher harmonics)
- Can use PWM. The speaker only has so fast a response, adds sort of like an average. So the average of the PWM output can approximate other waveforms.

Advanced music – FM synthesis

- So far have been doing AM (amplitude-modulation) by modifying the amplitude of the sine wave
- Can do FM (frequency modulation) where you rapidly change the frequency
- 1980s synthesizers and DOS sound cards (OPL2 based Soundblaster)

Advanced music – Digital Music / DAC

- Sample with ADC at some rate, maybe 44.1kHz
- Store the 16-bit samples
- Play back exact samples with DAC
- Really good playback. What's the downside? lots of disk space. 44k*2bytes (16-bits)*2-channel (stereo) = 160k/second
- Often use DMA so music is more in background. Often two DMA buffers, load one while other plays

Advanced music – Aside on Compression

- WAV – No compression, header on top of raw samples
- RLE – run length encoding?
- LZW – with dictionary
- MP3 – wavelets. Was patent encumbered until recently so had to pay royalties for encoding/decoding
- OGG Vorbis – free alternative to mp4
- FLAC – Free Lossless Audio Codec
- AAC – Advanced Audio Coding – Lossy

Low-memory music playing

- Even with MP3 compression, assuming our board was fast enough to play it, 1MB flash only enough for a minute or so of high-quality stero sound
- You could lower quality, but are there other ways to get music?

Low-memory music playing

- For older systems you can use something called a "tracker" that looks sort of like a spreadsheet, and you put in a list of notes to play (plus length, and effects)
- They have patterns, which can repeat (such as refrains)
- Notes/instruments use lookup tables, so it can be fast

AMIGA Mod Format

- Tracker
- 4 channels at once
- Short samples of instruments
- Tracker tellls you how to scale note of instrument, any effects to add
- 4 channels mixed
- Was in hardware, but can also do in software
- Reasonable size depending on how complicated/number of samples

AY-3-8910 Sound Chip

- ZX-Spectrum 128, Atari ST, arcade machines, Apple II Mockingboard
- 3 channels of square waves, can add noise
- Can set hardware envelope
- volume, frequency for each channel ABC
- 13 8-bit music registers (not all bits used)

YM Music format

- Just register dump of 13 registers at 50Hz
- 2 minute song 78k, not sound bad, but a lot for 8-bit machine with 64k RAM
- Compresses well, especially if you interleave so all registers (say all R13) together
- Old machines can be slow to decompress though
- Player is trivial, just 50Hz IRQ routine that loads 13 registers

PT3 Format

- Vortex Tracker
- Tracker format, list of patterns, notes, effects
- Music file usually less than 4k, can play in place
- Player more complex, 2k of code that calculates frequencies and volumes and effects
- Can fit player and many many songs in 1MB
- Wrote one for Apple II, convert code written for z80 processor and Pascal, comments in Russian

Advanced Player (Demo)

- Emulates an AY-3-8910 chip in software
- Talks to the audio codec on the STM board, allowing output to headphone jack
- This was way more trouble than it was worth
- Can hold a decent amount of songs in 1MB of flash though