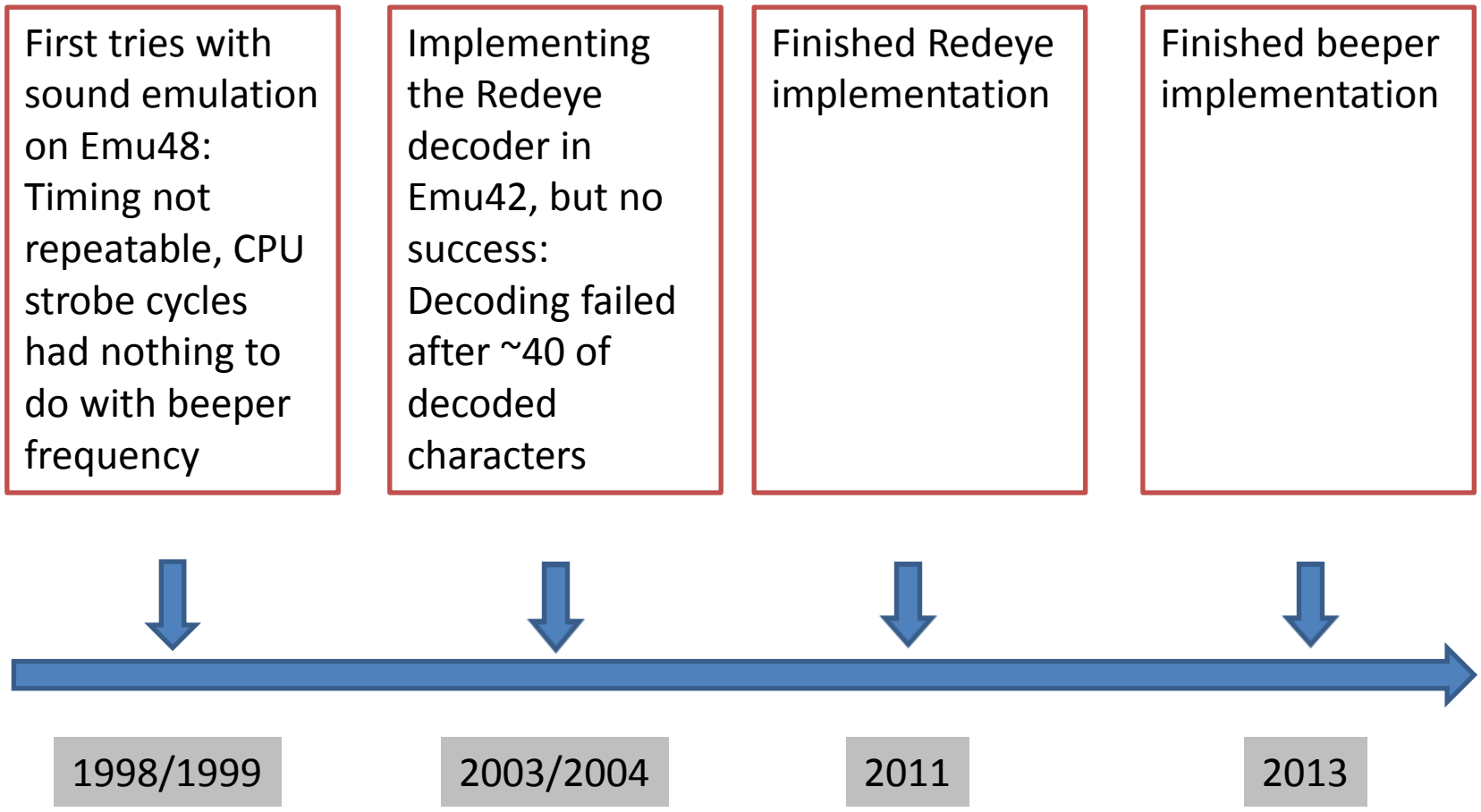
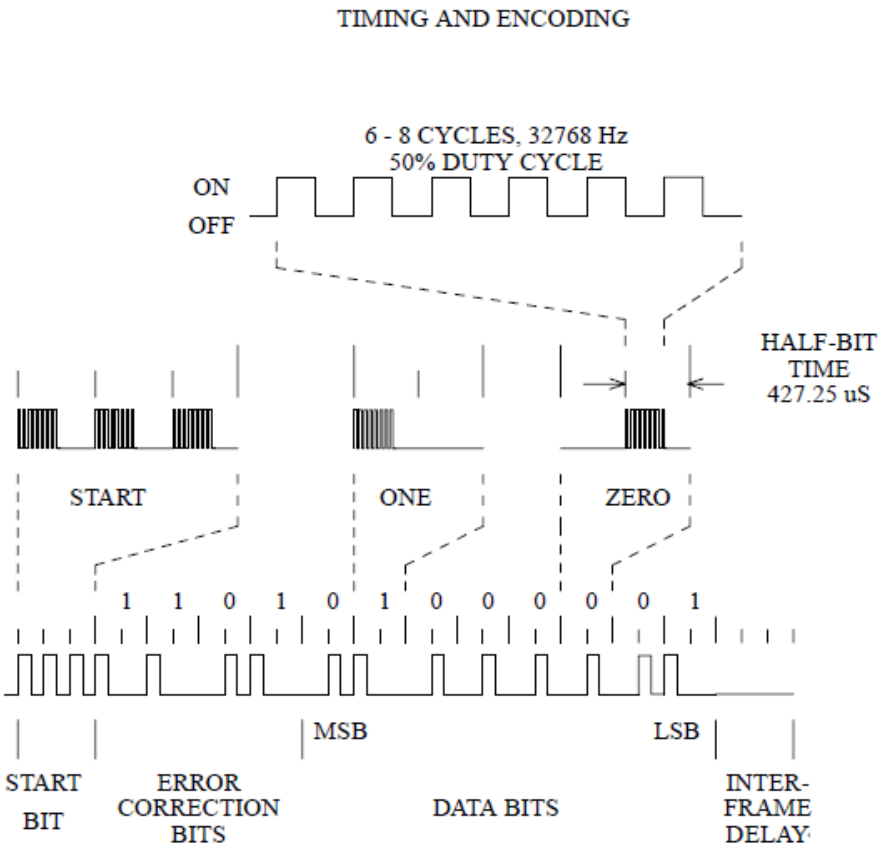
Long Term Development

Nearly 10 year‘s without sound and Redeye simulation inside the Saturn based emulators.

Timetable


First tries with sound emulation on Emu48: Timing not repeatable, CPU strobe cycles had nothing to do with beeper frequency
Implementing the Redeye decoder in Emu42, but no success: Decoding failed after ~40 of decoded characters
Finished Redeye implementation
Finished beeper implementation
1998/1999
2003/2004
2011
2013

Redeye Timing



- 32kHz Bursts generated by Hardware
- All other Timing generated by CPU

Problem on Calculators

- No hardware support for Redeye timing on Lewis CPU based hardware
- No hardware support for buzzer timing
- Timing must be generated by CPU core
- CPU strobe frequency not crystal driven, so timing is inaccurate

Solution on Calculators

- Measure the CPU speed with a crystal driven 512Hz (HP-71B) or 8192Hz timer
- Creating timing tables for buzzer and Redeye interface base on the measured CPU speed

CPU frequency measurement in detail

- Frequency f = $\frac{n}{t}$ where n is the no. of times an event occur and t is the time

  In the case of the calculator we measure the number of CPU loops (events) in a 62.5ms time frame.

Calculate the loops

The loop counter is implemented in a 3 nibbles register (12 bit) with 4095 counts maximum.

The algorithm simply count the number of software loops in a 62.5ms time frame.

The Code (HP-42S)

```
024E7   B=0     A
024E9   D0=(5)  #403F9          512Hz = 2ms counter
024F0   INTOFF                  keyscan disable
024F4   A=DAT0  B
024F7   C=DAT0  B
024FA   ?C=A    P, GOYES [024F7]  timer changed -> start measure
024FF   P=      1               C[B] = C[B] - #20
02501   C=C-1   P
02504   C=C-1   P
02507   B=B+1   X               inc measure counter          [6]
0250A   A=DAT0  B               reached end value (1/16Hz later)  [15]
0250D   ?C#A    B, GOYES [02507]  no, inc counter            [15]
                                B[X] speed factor
```

Calculate the frequency

From internal documents we know the strobe cycles for each opcode and so the overall strobe cycles for one software loop. This multiplied with the number of elapsed loops plus some corrections is the CPU strobe frequency in Hz.

This value divided by 16 is stored in the 5 nibble =CSPEED variable.

Relevance for Emulation

- No relevance for emulation with orignal speed
- But loop counter overflow for emulation with maximum speed

In this special case the loop counter overflow is something like returning a random number for the loops and so a random number for the CPU speed measurement.

Proper Timing

- All frequency calculations inside the calculator firmware return only proper results when the measured CPU strobe frequency is in a small range near to the nominal speed.
- All of my emulators recognize the speed measurement condition at the timer read access pattern and slow down emulaton speed during this measurement.

Redeye decoding

- The Redeye simulation capture the output enable bit of the 32kHz burst signal generator. With the elapsed CPU cycles the timing and so the sent character is reconstructed.
- The reconstructed character is send to a printer simulation over the UDP protocol.

Sound decoding

- The Sound emulation capture the output bit(s) connected to the buzzer. With the help of the elapsed CPU cycles since last output bit(s) change, the buzzer frequency is reconstructed.
- With a small delay of 60ms the collected frequency information is replayed over the Windows sound system interface.

"What a depressingly stupid machine."

Marvin