**Fred Hardware Specification**

This gives the considered design of “FRED” which is a mixture of known data from RCA documents, extrapolated deduction and (a little) pure guesswork. Fred appears to be derivative of System 00 (some sources describe the CPU as “FRED”)

CPU

The CPU is a TA6889/90 RCA1801 type (e.g. the reduced 59 instruction set) clocked at 1.0 Mhz as described in the 1975 Cosmac Microprocessor users manual.

*The speed of the processor is unknown, other than it has to be more than 504Khz (because there has to be sufficient time for 8 DMA Cycles in two NTSC Scan lines) and it has to be less than 4Mhz, because a 1802 can’t go any faster, and probably less than 2Mhz because 2102 SRAMs had a 500ns access time.*

*Because timing is unknown (it is probably 1Mhz-2Mhz) any timing critical stuff should be done by timers driven by the 60Hz interrupt, the only known constant time. The speed is deliberately set low so that if it is quicker time is just wasted.*

*It is logical that the design would be very close to the 1801/6889 design. This is supported by an article in the RCA annual summary which describes the LSI Fred 2 as being “repackaged in LSI form”. There is very little that could be removed from a 1801 instruction set without the result being unusable. (One of the subtracts could go, and perhaps shrc ?)*

*Clock speed should make allowances for 1 DMA cycle (8 clocks) for every displayed pixel (potentially), e.g. a 64x32 pixel will have 8x32 = 256 DMA reads, or 2048 clocks per frame. As the clock speed is unknown it’s a bit of a moot point anyway.*

Memory

Memory consists of 1k x 8 Static RAM, expandable to 4k x 8. Probably the Intel 2102 SRAM.

*For simplicity it is assumed this is partial decoding, e.g. 64 (or 16) mirrored copies of the RAM, this is not known. This should not however be used as a “feature”. There is a possibility of ROM in the system, such is described in the 1975 RCA summary “permanently resident general purpose subroutines”*

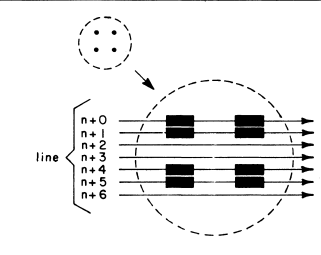
Input

All input devices, Keyboard, Tape and Punch Reader operate via DMA in in load mode. The keyboard can operate either in the double entry mode used in the Elf, or the shift on bit 4 + lower hex digit mode described in the Fred document. This can also be used to read the punch and tape (theoretically).

*The documentation says that “normally” the keyboard presses are written to RAM for each depression. This must be suppressed during video on, obviously, otherwise keyboard presses will overwrite pixel memory and distort the position of R0. It must also be suppressed during video blank for the same reason. We do not have the control of the display a 1861 gives us, it is much more autonomous.*

*Data is read at 30 bytes/second from the tape player, about the same as a guess from the gravity driven punch reader, and much more slowly from the keyboard. Therefore input could be delayed by a flip/flop action until the end of the video page.*

*At interrupt, if a keystroke has been received R0 will be set to the end value of the display (e.g. have 128 or 256 added to its current value). If a keystroke has been received then this will be stored at (R0) (e.g. the byte after the end of the display RAM) in the 000<shift><hex> format (for this purpose the double-byte entry will not operate in run mode), and R0 will be incremented. This increment (normally bit 0 set) will allow the detection of a key. Note this will not occur on the first interrupt (as R0 may not have been initialised at that point). If this is complete tosh and it is done via Port 4 as in the ELF (say) this can be changed to do it this way. It is theoretically possible that two key presses could take place in the same frame. Consideration could be given to using half a register to store the key stroke ($FF = no key) and the other half could be the high byte of video RAM. Thus the video IRQ is all internal.*

Video Hardware

An interrupt occurs every frame, at the beginning (e.g. at 60Hz).

There is, I think, no requirement for coding techniques other than (i) the processor to set R0 to the top of the display frame at this time and (ii) this should be done pretty much immediately in the interrupt routine before video generation starts. R0 should be unmodified throughout the frame, though of course in reality it would go up 4 or 8 every dot line when the display system does 4 or 8 DMAOUTs into its buffer, so it needs to be reset at Interrupt time.

This produces 16 or 32 display lines, each of four scan lines. The first two scan lines are dot information and consist of 32 or 64 dots spaced 50/50, according to the mode settings.

*It is a logical deduction that there is no interrupt routine requirement other than setting R0 for DMA out. All buffering and timing is done by the video hardware, which DMAOUTs data into its 8 byte buffer.*

*Additionally, an RCA document talking about the 1802 in general describes a “dot video display” that has an Interrupt routine that “sets the pointer”. The video data is known to be buffered, so there is no 1861-style requirement to sync data out at a specific time.*

*It is not known how 64x32 32x32 64x16 modes are selected, so this is done via a macro, which writes a constant set as two defines (columns and rows) to a chosen port. The documentation does not mention a 32x16 one which should be theoretically possible so this is not permitted.*

*It is not known how Interrupts are enabled or disabled, or even if this is possible. They are initialised in the standard initialisation code or would if it becomes known it is required. Disabling is not allowed, in case it isn’t possible.*

*We assume we aren’t doing any smart 2600-style tricks and changing the video hardware or pointer during video generation. There wouldn’t be much point anyway.*

Sound Hardware

A 6889/1801 has no Q line or equivalent, so it is therefore assumed that the sound is connected to a output port of some sort. Note that keyboard presses also cause a beep to occur, the pitch varying with the hex value pressed. Exactly how this works is not known.

*This is handled by a separate routine which toggles a port bit, assumed to be the same port as the video controller at a given rate.*

*Pitches should be defined using a macro which does an LDI with a scalable value (so that the pitch can be changed as the frequency changes) which then calls a system routine, which is driven via a timer connected to the interrupt.*

Other

The relay control output latch is not emulated. The Detect tape leader/trailer is not emulated. I do not plan to allow this, though it could be done using WAV files or similar.

One presumes the “girls” picture of Fred 2 from the Sarnoff collection, has the gravity punched card reader back left, control switches front left, hex keypad front right, and possibly the speaker at the rear right ?