Intro to CRTC Registers on the BBC Micro (part #1)

Kieran Connell

vABUG Masterclass

14/01/2021

Disclaimer: this is all somewhat simplified, far more detailed explanations can be found online!

See New Advanced User Guide, pp. 187
See https://en.wikipedia.org/wiki/Motorola_6845

1. What can we do with the CRTC registers?

1. Custom modes / change screen dimensions to save RAM
2. Hardware scrolling!
3. “Vertical rupture” effects
4. Smooth vertical scrolling

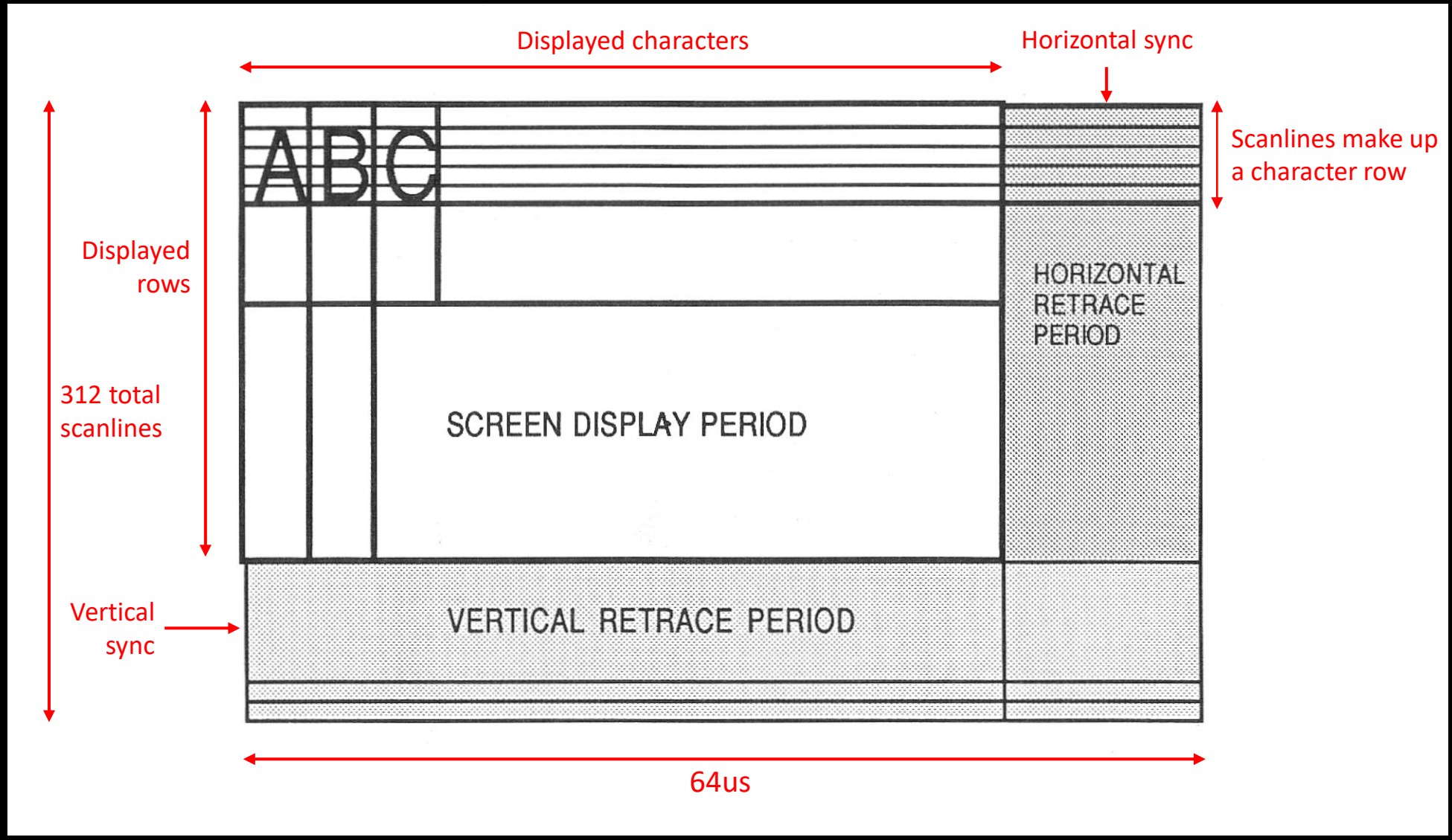
For part 2 (sorry!)

2. What is the CRTC responsible for?

For another Masterclass!
(Or check out
http://beebwiki.mdfs.net/Video_ULA)

6845 CRTC **“Display controller”**	**Acorn Video ULA** **“Video processor”**
• Generates signals to control a raster display → Vertical sync → Horizontal sync	• Generates the pixels sent to the display
• Generates correctly timed address values for screen RAM to be displayed	• Reads bytes from those RAM addresses • Converts bytes to pixels through palette
• Also manages interlace, cursor, lightpen etc.	• Controls CRTC clock rate (1MHz/2MHz) • Controls output pixel rate (2/4/8/16MHz)
• 18x write-only registers • Accessed through SHEILA &FE00 and &FE01	• 1x write-only control register • 64-bits of palette RAM (16 x 4 bits) • Accessed through SHEILA at &FE20 and &FE21

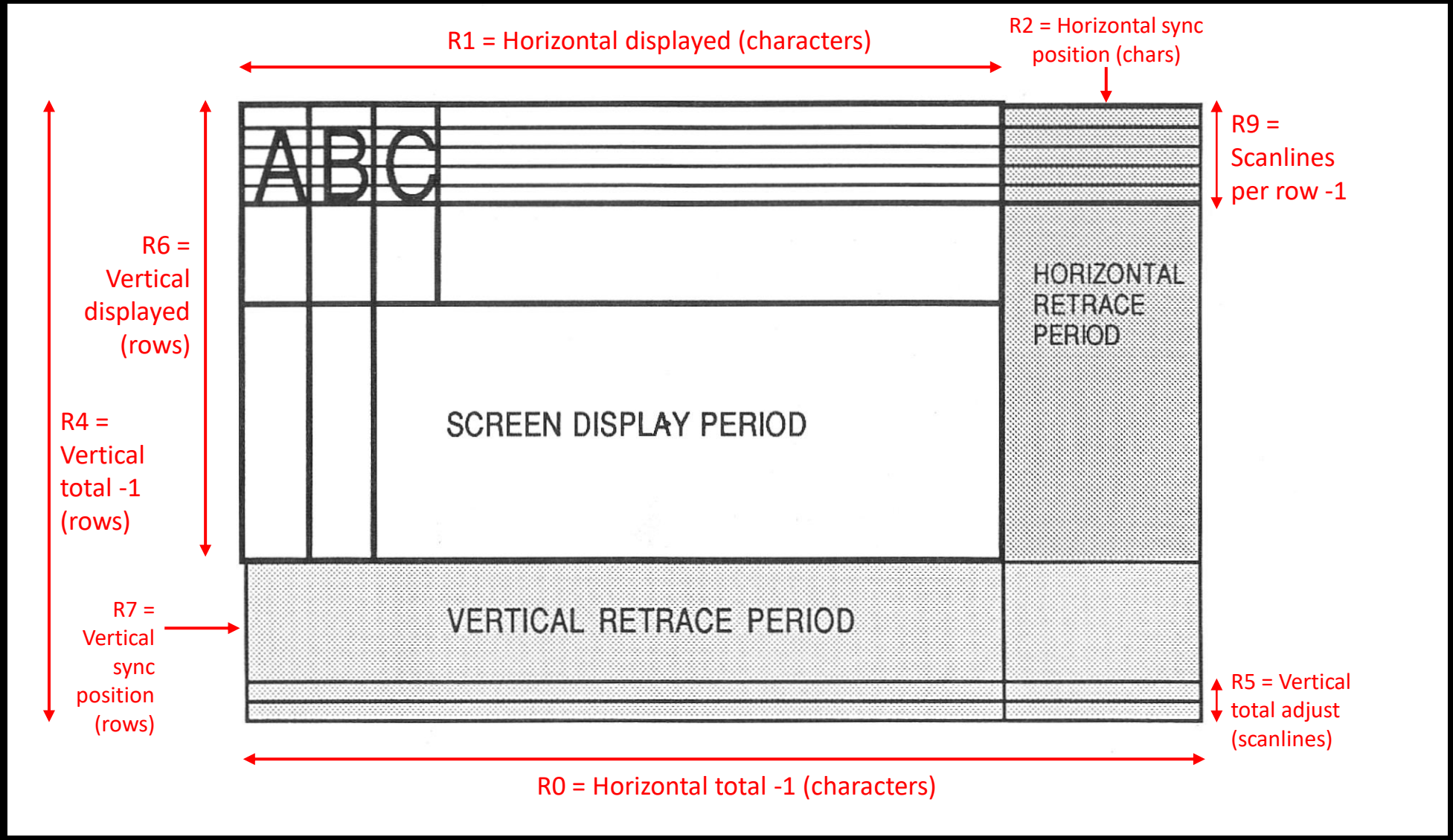
3. 6845 CRTC display generation



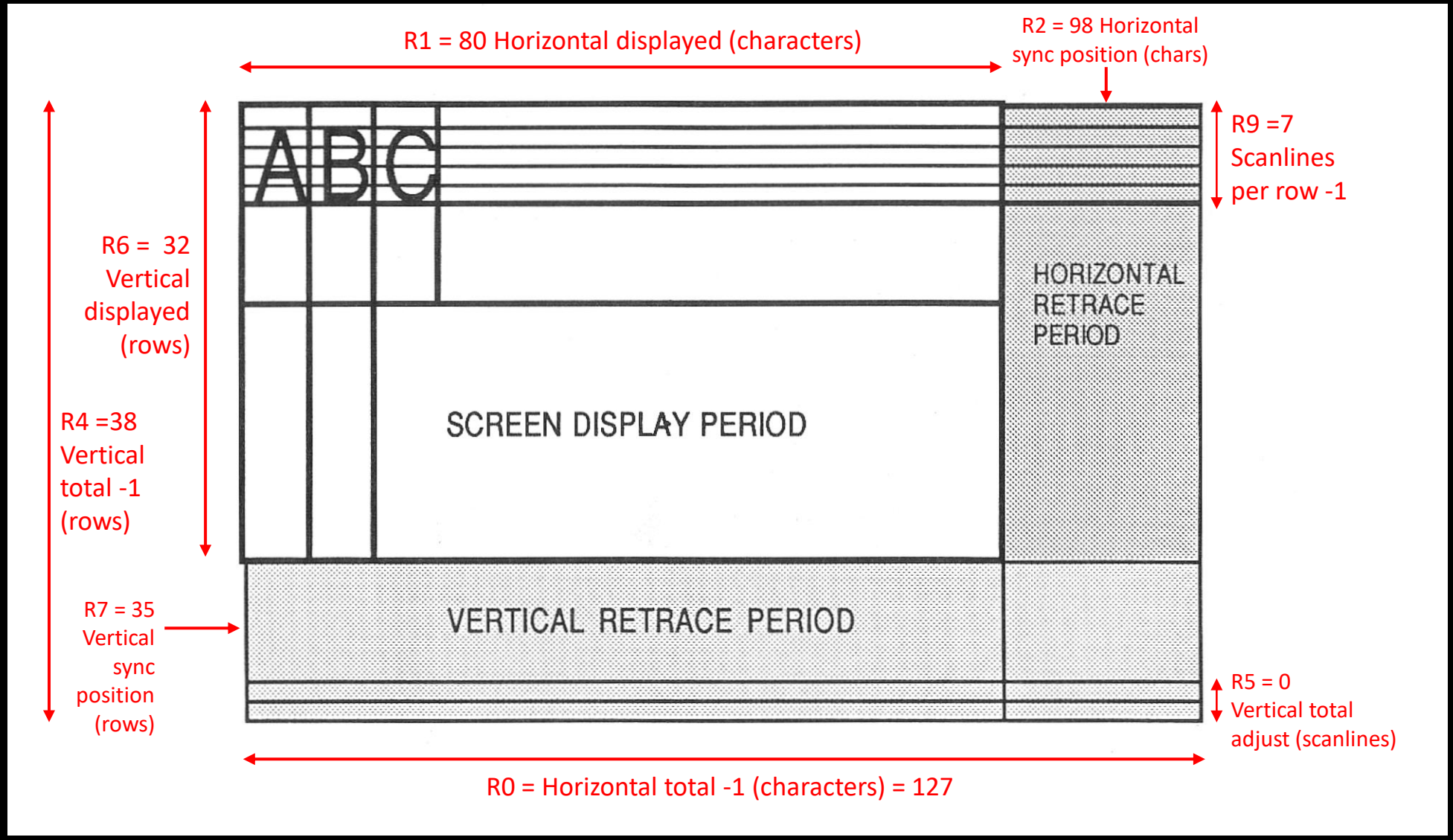
4. 6845 CRTC Registers

6845 Register	Function	Units
R0	Horizontal total (-1)	(characters)
R1	Horizontal displayed	(characters)
R2	Horizontal sync position	(characters)
R3	Horizontal sync width + Vertical sync time	(bit fields)
R4	Vertical total (-1)	(rows)
R5	Vertical total adjust	(scanlines)
R6	Vertical displayed	(rows)
R7	Vertical sync position	(rows)
R8	Interlace mode + Display delay + Cursor delay	(bit fields)
R9	Scanlines per character row (-1)	(scanlines)
R10	Cursor start + Cursor type + Cursor blink	(bit fields)
R11	Cursor end	(scanlines)
R12, R13	Display start address (high, low)	(character address)
R14, R15	Cursor position (high, low)	(character address)
R16, R17	Lightpen position (high, low)	(character address)

5. 6845 registers mapped onto the display



6. 6845 registers for MODE 0/1/2 display



7. Programming the 6845

```
; The official way via VDU 23 code 0
VDU 23;REGISTER,VALUE;0;0;0
; Poke registers directly in BASIC
?&FE00=REGISTER : ?&FE01=VALUE
; Poke registers directly in assembler
LDA #REGISTER : STA &FE00
LDA #VALUE : STA &FE01
; All register values take effect immediately*
```

Demo: 6845 Register Browser

8. Display start address R12/R13

- 6845 works on characters: 1 character = 8 bytes
- CRTC character addresses are (RAM addresses DIV 8)
- Can be anywhere in the bottom 32K of the memory map: RAM &0000 - &7FFF → CRTC &0000 - &0FFF
- Display start address R12= High byte, R13= Low byte
- These values are *Latched*. They are only read at the start of a new display ‘cycle’.
- Normally one display cycle per display frame…
  → vertical rupture (next time!)

9. Hardware scrolling

- Set new display start address each frame into R12 & R13. Easy! So in MODEs 0,1,2:
- +1 to start address moves display *left* 1 character
- -1 to start address moves display *right* 1 character
- +80 to start address moves display *up* 1 row
- -80 to start address moves display *down* 1 row
- Hardware scrolling is coarse but fast!

10. What happens above &8000?

- Address wraparound logic kicks in above &8000:

Latch value	Amount subtracted	Restart address	Modes	Screen size
0	&4000	&4000	3	16K
1	&2000	&6000	6	8K
2	&5000	&3000	0, 1, 2	20K
3	&2800	&5800	4, 5	10K

- Set via the addressable latch, see NAUG pp.386*
- (Or copy the code from the HW scrolling demo.)
- See http://beebwiki.mdfs.net/Address_translation for full details**

Demo: Hardware scrolling

Questions?

https://bitshifters.github.io/

See also the Twisted Brain demo write up on Stardot:
https://stardot.org.uk/forums/viewtopic.php?f=53&t=15300