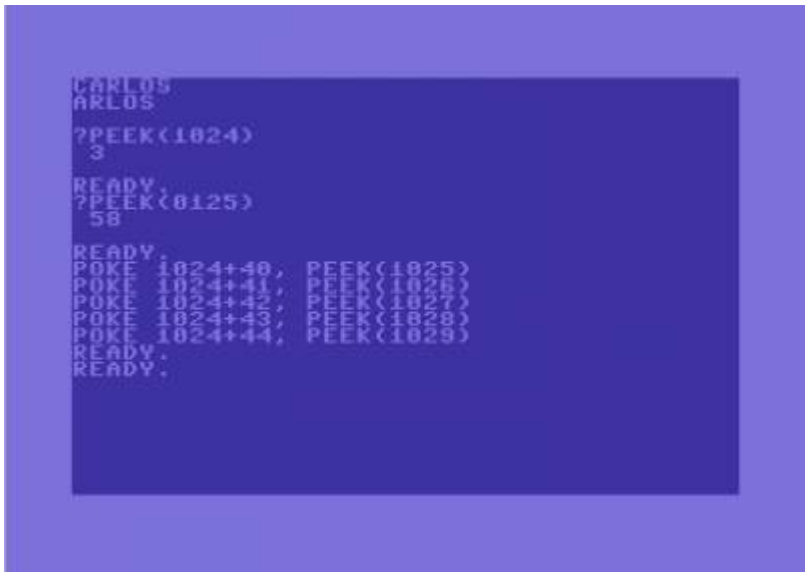
Text scrolling with the VIC

By Monte Carlos

Introduction

Text scrollers are done by moving one or more lines of the screen left or right and adding new characters at the border. They are using the simplest mode the grafic chip of the c64 offers, namely the text mode. See http://codebase64.org/doku.php?id=base:built_in_screen_modes for more information on the c64 grafics modes.
A memory area is associated with the textscreen, the grafic chip displays. (The memory area for the textscreen lies at $0400(1024) after switching on the computer and has 1000 bytes) A scroller can be done by simply peeking a value from the textscreen and poking this value at the byte associated with the neighbour character and doing this consecutively for the whole textline.

Example:



Scrolling

Hardscroll in one direction

In the following example, the text is scrolled left and a new character is inserted at the rightmost screenposition.

```
screen = $0400

  ldx #38
  ldy #0
scroll:
```

```
  lda screen+1,y
  sta screen,y
  iny
  dex
  bpl scroll

  lda newchar
  sta screen+39
```

Next example is the code for moving text left to right.

```
screen = $0400

  ldy #38
scroll:
  lda screen,y
  sta screen+1,y
  dey
  bpl scroll

  lda newchar
  sta screen
```

A common pitfall is to move the bytes in the wrong order. Say you want to scroll the string **commodore**. Then, after the transfer, it looks like **ommodore** if you move first the o, then the first m, then the second m etc. Contrary you get **eeeeeeeee** if you start moving the e, then the now overwritten r, then the now overwritten o.

Bigger scrollers

See Bigger letters for information about how to assemble characters to bigger letters in a way so that each character represents only a part of the letter. The only difference to 1×1 text scroller like described above is, that you have to scrolll more than one line.
For example scrolling n lines looks like this:

```
screen = $0400

  ldx #38
  ldy #0
scroll:
  lda screen+1,y
  sta screen,y
  lda screen+41,y
  sta screen+40,y
  ...
  lda screen+((n-1)*40)+1,y
  sta screen+((n-1)*40),y
  lda screen+(n*40)+1,y
  sta screen+(n*40),y
```

```
  iny
  dex
  bpl scroll

  lda newchar
  sta screen+39
  lda newchar+1
  sta screen+40+39
  ...
  lda newchar+n-1
  sta screen+((n-1)*40)+39
  lda newchar+n
  sta screen+(n*40)+39
```

The charset memory may not contain latin symbols. It can also hold parts of scenery for games. The characters then represent small parts of the background which is build up by these characters like a puzzle. This way the scenery can be easily scrolled like text.

Bidirectionall hardscroll

Of course it is possible to write extra code to scroll left and to scroll right. (See the examples above) But this isn't necessary, if we use speedcode for the textscrolling and the y register to temporarily save the characters at the destination positions (Speedcode basics: See article by Cruzer/CML:speedcode). Look at the following example:

```
lda screen,x
sty screen
ldy screen+1,x
sta screen+1
lda screen+2,x
sty screen+2
...
```

If the routine is started with x=0, the text is scrolled right. If the routine if started with x=2, the text is scrolled left. Is is also possible to use exactly the same routine for 4 or 8 ways scrolling (See article and homepage by Malcolm Bamber: 4_ways_scroll,http://www.dark-well.pwp.blueyonder.co.uk/). Care must be taken for the characters at the screen borders. Extra code has to be added for each direction to supply the new characters scrolling into the visible area. This extra code is much smaller than the core routine and so we save approximately 7/8 of the memory compared to 8 scroll routines for each direction. If we do not use speedcode for the scrolling, things get a little more complicated and the stack has to be used to temporaly save characters which may become overwritten.

```
        LDX #39
        LDY #39; for left scrolling
        LDX #39
        LDY #37; for right scrolling

        LDA SCRLADR,Y
        STA TMP1
```

```
          LDA SCRLADR-1,Y
          PHA
S
          LDA TMP1
          STA SCRLADR-1,X
          PLA
          STA TMP1
          LDA SCRLADR-2,Y
          PHA
          DEY
          DEX
          BNE S
          PLA
```

Softscroll

Role of $d016

The code above enables you to scroll a text in 8 pixel steps. If you want to do a smooth scroll two pixel steps or 1 pixel steps are desirable. This is done using the soft-scroll register $d016 (for a detailed description look: https://sh.scs-trc.net/vic/). If you want to scroll left you start with $x7 in $d016 and decrease that value every frame. After 8 times you reset this register to its start value and move all characters one byte backwards using the according one of above scroll routines.

Example code

What follows is the perhaps shortest routine for bidirectional soft scrolling (only $6d bytes long), from which above code is taken from:

```
          *= $1000

Q         = 2
XPIXSHIFT = 4
TMP1      = 5
TEXTADR   = 6

SCRLADR   = $0400

          JSR $E544

          SEI
TEXTRESTART
          LDA #<TEXT
          STA TEXTADR
          LDA #>TEXT
          STA TEXTADR+1
```

```
LOOP     INC $d012
         BNE LOOP

DESTSTART = *+1
         LDX #39;39
SRCSTART = *+1
         LDY #39;37

XPIXSHIFTADD
         DEC XPIXSHIFT

         LDA XPIXSHIFT
         AND #7
         STA $D016

         CMP XPIXSHIFT
         STA XPIXSHIFT
         BEQ LOOP

         LDA SCRLADR,Y
         STA TMP1
         LDA SCRLADR-1,Y
         PHA
S
         LDA TMP1
         STA SCRLADR-1,X
         PLA
         STA TMP1
         LDA SCRLADR-2,Y
         PHA
         DEY
         DEX
         BNE S
         PLA
GETNEWCHAR
;TEXTADR  = *+1
         LDA (TEXTADR,X)
         BEQ TEXTRESTART

         INY
         BMI *+4
         LDX #$27

NOBEGIN  INC TEXTADR
         BNE *+4
         INC TEXTADR+1

         TAY
         BMI DIRCHANGE
```

```
         STA SCRLADR,X
         BPL LOOP
;--------------------------------------
DIRCHANGE LDA XPIXSHIFTADD
         EOR #$20
         STA XPIXSHIFTADD

         LDX DESTSTART
         LDY SRCSTART
         DEX
         INY
         STX SRCSTART
         STY DESTSTART
         BNE LOOP
;--------------------------------------
TEXT     .TEXT " THIS SCROLLER CAN"
         .TEXT " SCROLL IN FORWARD"
         .TEXT " AND BACKWARD DIREC"
         .TEXT "TION!               "
         .TEXT "                    "
         .TEXT "         "
         .BYTE $FF
         .TEXT "WON GNILLORCS MORF "
         .TEXT "TFEL OT THGIR ... . "
         .TEXT "                    "
         .TEXT "                    "
         .BYTE $FF,0
```