Subject: Why does this invert the whole screen?
Posted by zellyn on Tue, 04 Apr 2017 13:40:02 GMT
View Forum Message <> Reply to Message

10 NORMAL
20 PR#3
30 INVERSE
40 PRINT "*"
50 NORMAL

If you print something in "normal" mode before inverting, it does what you'd expect.

Zellyn

Subject: Re: Why does this invert the whole screen?
Posted by gids.rs on Tue, 04 Apr 2017 15:53:17 GMT
View Forum Message <> Reply to Message

On Tuesday, April 4, 2017 at 7:40:03 AM UTC-6, Zellyn wrote:
> 10 NORMAL
> 20 PR#3
> 30 INVERSE
> 40 PRINT "*"
> 50 NORMAL
>
> If you print something in "normal" mode before inverting, it does what you'd expect.
>
> Zellyn

PR#3 temporarily takes output control away from the OS by changing the output vectors at $36.37, and puts control in applesoft. A PR#3 by itself does not initialize the 80-col firmware, it just sets the vectors at $36.37 for the next output character. The 80-col firmware is initialized when the first character is printed and initializing the 80-col firmware also clears the 80-col screen but to the mode set (which is inverse), but first prints the first character it sees.

In machine language, I usually initialize the 80-col firmware with the 80-col normal command, which if you know the Control codes for 80-col is (Ctrl-N is 14 and Ctrl-O is 15 for normal and inverse).

```
LDA #14    ; normal mode in 80-col firmware, disables inverse and mousetext
JSR $C300
```

If you were to replace the inverse command to 30 ? CHR$(15) and normal command to CHR$(14), all would be normal.

You could combine lines 30, 40 and 50 to this to work normally:

30 ? CHR$(15)"*"CHR$(14)

Using the OS command CHR$(4) takes care of the initialization of the 80-col firmware. The syntax for line #20 would then be:

20 ? CHR$(4)"PR#3"

Subject: Re: Why does this invert the whole screen?
Posted by Anonymous on Tue, 04 Apr 2017 16:04:51 GMT
View Forum Message <> Reply to Message

Originally posted by: John Brooks

On Tuesday, April 4, 2017 at 8:53:18 AM UTC-7, gid...@sasktel.net wrote:
> On Tuesday, April 4, 2017 at 7:40:03 AM UTC-6, Zellyn wrote:
>> 10 NORMAL
>> 20 PR#3
>> 30 INVERSE
>> 40 PRINT "*"
>> 50 NORMAL
>>
>> If you print something in "normal" mode before inverting, it does what you'd expect.
>>
>> Zellyn
>
>
>
> PR#3 temporarily takes output control away from the OS by changing the output vectors at $36.37, and puts control in applesoft. A PR#3 by itself does not initialize the 80-col firmware, it just sets the vectors at $36.37 for the next output character. The 80-col firmware is initialized when the first character is printed and initializing the 80-col firmware also clears the 80-col screen but to the mode set (which is inverse), but first prints the first character it sees.
>
> In machine language, I usually initialize the 80-col firmware with the 80-col normal command, which if you know the Control codes for 80-col is (Ctrl-N is 14 and Ctrl-O is 15 for normal and inverse).
>
> LDA #14 ; normal mode in 80-col firmware, disables inverse and mousetext
> JSR $C300
>
>
> If you were to replace the inverse command to 30 ? CHR$(15) and normal command to CHR$(14), all would be normal.

>
> You could combine lines 30, 40 and 50 to this to work normally:
>
> 30 ? CHR$(15)"*"CHR$(14)
>
>
> Using the OS command CHR$(4) takes care of the initialization of the 80-col firmware. The syntax for line #20 would then be:
>
> 20 ? CHR$(4)"PR#3"

Note that ASCII 14/15 requires 80-col firmware which is only present on the //e,//c, & IIGS. I prefer the last option: CHR$(4)"PR#3" as it works on all 80-col displays including Apple][&][,][+, //e, //c, or IIGS has an 80 column card available:

```
 lda $BF98
 bit #2
 bne Has80Col
```

-JB
@JBrooksBSI

Subject: Re: Why does this invert the whole screen?
Posted by zellyn on Tue, 04 Apr 2017 20:01:51 GMT
View Forum Message <> Reply to Message

I love this group. Thanks folks!

ps. Of *course* PR#3 doesn't do anything until you print. D'oh!

Subject: Re: Why does this invert the whole screen?
Posted by Anonymous on Tue, 04 Apr 2017 22:05:26 GMT
View Forum Message <> Reply to Message

Originally posted by: Brian Patrie

On 2017-04-04 10:53, gids.rs@sasktel.net wrote:
> Using the OS command CHR$(4) takes care of the initialization of the
> 80-col firmware. The syntax for line #20 would then be:
>
> 20 ? CHR$(4)"PR#3"

I read that as saying that making this a deferred OS command initializes the 80-col firmware before emitting the first char. My jaw kinda dropped; then i tested it, and this does not seem to be the case. The programme exhibited the same behaviour--i.e. inverted the entire screen,

due to inverse mode being in effect when the 80-col firmware initialized.

Sorry if i merely misunderstood. (But then i might not be the only one.)

Subject: Re: Why does this invert the whole screen?
Posted by gids.rs on Wed, 05 Apr 2017 02:35:10 GMT
View Forum Message <> Reply to Message

On Tuesday, April 4, 2017 at 4:05:29 PM UTC-6, Brian Patrie wrote:
> On 2017-04-04 10:53, Rob wrote:
>> Using the OS command CHR$(4) takes care of the initialization of the
>> 80-col firmware. The syntax for line #20 would then be:
>>
>> 20 ? CHR$(4)"PR#3"
>
> I read that as saying that making this a deferred OS command initializes
> the 80-col firmware before emitting the first char. My jaw kinda
> dropped; then i tested it, and this does not seem to be the case. The
> programme exhibited the same behaviour--i.e. inverted the entire screen,
> due to inverse mode being in effect when the 80-col firmware initialized.
>
> Sorry if i merely misunderstood. (But then i might not be the only one.)

You are right. I assumed the DOS/PRODOS command of chr$(4) actually initialized the 80-col screen since that was the standard practice. Both Dos's just manipulate the input/ouput hooks ($36.39) for their own use. I should have checked this scenario as well.

The same till holds true though with the 80-col firmware being initialized just before the first character is printed to the screen.

The INVERSE command changes the text output format variable $32 of the zero-page to $3F. Normal mode this value is $FF.

Before the first character is printed to screen, the screen is cleared and the cursor moved to the HOME position, and if the variable $32 is set to $3F with an INVERSE command, then the entire screen will become inverse.

You can see this with a poke instead

```
10 PR#3
20 POKE 50,63  ; 50 is the decimal value of $32 and 63 is the dec val of $3F
30 ?"ABCDEF"
40 NORMAL
```

The solution is still the same and that is to print a control character before turning on the

INVERSE command.

In my original example I used the value 14 for the chr$ value which is an 80-col control character command for normal mode. But any of the control characters can be used.

This next example will print the mousetext characters but does not inverse the rest of the screen.

```
10 PR#3
20 ? CHR$(27);:  REM a semicolon here will suppress the <RTN> and prevent
30 INVERSE    :  REM starting printing on the 2nd line.
40 ?"ABCDEFGHIJKLMNOPQRSTUVWXYZ"
50 NORMAL
```

On an Apple II+ you can use the null character of CHR$(0)

```
10 PR#3
20 ? CHR$(0);
30 INVERSE
40 ?"*"
50 NORMAL
```

Control character actions in 80 col mode on a IIe/IIc/IIGS

CTRL HEX CHR$
@ ($80) 0 - NULL character
A ($81) 1 -
B ($82) 2 -
C ($83) 3 -
D ($84) 4 -
E ($85) 5 -
F ($86) 6 -
G ($87) 7 - bell
H ($88) 8 - left arrow
I ($89) 9 - tab key
J ($8A) 10 - down arrow (linefeed)
K ($8B) 11 - clears from cursor to end of screen
L ($8C) 12 - homes the cursor and clears the window
M ($8D) 13 - Return key
N ($8E) 14 - Normal
O ($8F) 15 - Inverse
P ($90) 16 -
Q ($91) 17 - set display to 40 column mode
R ($92) 18 - set display to 80 column mode
S ($93) 19 - pause listing
T ($94) 20 -

U ($95) 21 - disables enhanced video firmware
V ($96) 22 - scrolls the display down one line while leaving the cursor in the same position
W ($97) 23 - scrolls the display up one line while leaving the cursor in the same position
X ($98) 24 - disables mousetext
Y ($99) 25 - homes the cursor without clearing the window
Z ($9A) 26 - clears entire line contaning cursor
[($9B) 27 - ESC key, enables mousetext character ROM
\ ($9C) 28 - move one space to the right
] ($9D) 29 - clears from cursor to end of line
^ ($9E) 30 -
_ ($9F) 31 - move cursor up one line