Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by Anonymous on Sun, 14 Nov 2021 04:00:00 GMT
View Forum Message <> Reply to Message

Originally posted by: nospam.Bernd.Bachmann

Hello George,

that bug you discovered on the Plus 4 kernel rom sounds quite interesting.
I still use the Plus 4 since many many years (actually it was my very
first 8bit computer I owned).

Maybe you are interested in posting this bug on the Plus4 website we have
at: https://plus4world.powweb.com/

There is a forum on this site were we could discuss this bug.
Feel free to join us there! :-)

Best regards,
Bernd (aka Fonsis on the Plus4world Site)

> I was going over some of my old CBM files from back in the day, and ran
> across an error I had found in the Plus 4 kernel rom. I don't think I ever
> .....
> about this, but it would be nice to make a record in case anyone ever
> wanted to make other revisions to the ROM, or actually make use of the UART
> capability of the Plus 4.

> George Hug
> -+- SoupGate-Win32 v1.05
> + Origin: Agency HUB, Dunedin - New Zealand | Fido<>Usenet Gateway
> (3:770/3)

Viele Gruesse
Bernd

Subject: Plus 4 rom error - is there any place to report it?
Posted by George on Sun, 14 Nov 2021 04:03:02 GMT
View Forum Message <> Reply to Message

I was going over some of my old CBM files from back in the day, and ran
across an error I had found in the Plus 4 kernel rom. I don't think I ever
found a way to report it to anyone, so I thought I would see if anything
has changed.

The error is in the 6551 ACIA servicing routine where a byte is read in
from the ACIA:

```
LDA $FD00
BEQ EAC2
STA $07D5
```

Incoming bytes are first stored at $0FD5, and later moved from there into
the input buffer. But as the rom is written, any null byte (00) received
would be later stored as whatever the most recent non-null byte was. And
it's impossible to receive a null byte. The solution is to reverse the
second and third instructions:

```
LDA $FD00
STA $07D5
BEQ EAC2
```

Or you could duplicate the beginning of the IRQ servicing up to this point
in your code, with the correction, then jump back into the rom.

Of course, this being the Plus 4, it may be that nobody would ever care
about this, but it would be nice to make a record in case anyone ever
wanted to make other revisions to the ROM, or actually make use of the UART
capability of the Plus 4.

George Hug

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by Jim Brain on Sun, 14 Nov 2021 15:52:21 GMT

View Forum Message <> Reply to Message

On 11/13/2021 10:03 PM, George wrote:
> I was going over some of my old CBM files from back in the day, and ran
> across an error I had found in the Plus 4 kernel rom. I don't think I ever
> found a way to report it to anyone, so I thought I would see if anything
> has changed.
>
> The error is in the 6551 ACIA servicing routine where a byte is read in
> from the ACIA:
>
> LDA $FD00
> BEQ EAC2
> STA $07D5
>
> Incoming bytes are first stored at $0FD5, and later moved from there into
> the input buffer. But as the rom is written, any null byte (00) received

> would be later stored as whatever the most recent non-null byte was. And
> it's impossible to receive a null byte. The solution is to reverse the
> second and third instructions:
>
> LDA $FD00
> STA $07D5
> BEQ EAC2
>
> Or you could duplicate the beginning of the IRQ servicing up to this point
> in your code, with the correction, then jump back into the rom.
>
> Of course, this being the Plus 4, it may be that nobody would ever care
> about this, but it would be nice to make a record in case anyone ever
> wanted to make other revisions to the ROM, or actually make use of the UART
> capability of the Plus 4.
>
> George Hug
>

I forwarded it the cbm-hackers mailing list, where a bunch of the
technical gurus hang out.

Jim

--
Jim Brain, brain@jbrain.com
RETRO Innovations: Contemporary Gear for Classic Systems
www.go4retro.com

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Sun, 14 Nov 2021 18:40:44 GMT
View Forum Message <> Reply to Message

Bernd Bachmann says...

> that bug you discovered on the Plus 4 kernel rom sounds
> quite interesting. I still use the Plus 4 since many
> many years (actually it was my very first 8bit computer
> I owned).

> Maybe you are interested in posting this bug on the
> Plus4 website we have at: https://plus4world.powweb.com
> /

> There is a forum on this site were we could discuss this
> bug. Feel free to join us there! :-)

Thanks Bernd, I just posted there. Hope it will be useful
to someone.

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Sun, 14 Nov 2021 19:12:04 GMT
View Forum Message <> Reply to Message

Jim Brain says...

> I forwarded it the cbm-hackers mailing list, where a
> bunch of the technical gurus hang out.

Thanks very much, Jim. It's too bad the +4 didn't get wider
use. The 6551 ACIA was a major improvement over the
horrendous emulation fiasco in the C64. I wrote replacement
code for the C64, but still only got it up to 2400 baud in
full duplex. The 6551 could I'm sure do 9600 baud, and
maybe 19,200. You just have to service one interrupt per
byte, and the 6551 does all the work.

While I'm here, is there any need for an ML Monitor program
for the C64, or has that already been done many times? Mine
is similar to the Monitor rom section found in the +4, but
is loaded into the C64 by a Basic loader, and you say where
you want it placed. My memory is it's exactly 2K (8 pages).
Comes with a manual. I think one difference between mine
and the +4 logic is that mine correctly moves overlapping
blocks in both directions, and the +4 didn't.

I also have an assembler that runs on the C64, but I assume
people just cross-assemble on their PCs these days.

And finally, I have an article I wrote for Transactor, but
they shut down before publishing it, which I think is the
final word on the REL file bug in the 1541. Includes the
1541 rom fix to eliminate it.

If there's any use for any of this, I could post it in a
Github repo.

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by Jim Brain on Mon, 15 Nov 2021 06:44:15 GMT
View Forum Message <> Reply to Message

On 11/14/2021 1:12 PM, George wrote:

> Jim Brain says...
>
>> I forwarded it the cbm-hackers mailing list, where a
>> bunch of the technical gurus hang out.
>
> Thanks very much, Jim. It's too bad the +4 didn't get wider
> use. The 6551 ACIA was a major improvement over the
> horrendous emulation fiasco in the C64. I wrote replacement
> code for the C64, but still only got it up to 2400 baud in
> full duplex. The 6551 could I'm sure do 9600 baud, and
> maybe 19,200. You just have to service one interrupt per
> byte, and the 6551 does all the work.

Indeed you can. You can also do 115200 on the IC by using the /16 Bps
clock override in the register config. The 6551A can do 38400 normal
and 230400 using the same /16 trick.

CBM Hackers responses:

Hm... The way I read the datasheet of the 6551, you need to check the
status register whether a byte is waiting (Bit 3 set) and if yes, grab
the byte and store it into the buffer. That BEQ doesn't really make
sense in this context.

Gerrit

If I remember well I have used the serial port as tty terminal in the
past and it was working fine (although probably that does not use a 0x0
byte). Also at the university a guy has written a SLIP protocol software
and could get IP packets. He has created telnet, ftp and it was working.
It was the subject of his thesis and he has graduated.

Istvan

Maybe you could provide a bit more context on your note...

I read your note that the receive routine will read the READ register of
the 6551 ($fd00) and then go elsewhere of the value is 0, storing it at
$0fd5 (though you also say $07d5, which confused me, maybe a typo?) if
<>0. The text, though, states that the routine will receive a null
byte, not store it, but then when a non-null byte is received, it will
store the null in the place the non-null byte was supposed to be stored.
That would seem to be a huge issue, and I'm not aware anyone sees such
behavior.

Am I reading your notes correctly?

Gerrit's comment above is noting that the BEQ doesn't make any sense, as

by the time the routine reads data from the READ register, it should
always have previously checked the data available flag. If set, the
data in the read register should be stored regardless, and no
conditional should be performed.

Jim

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Mon, 15 Nov 2021 15:50:41 GMT

View Forum Message <> Reply to Message

Jim Brain says...

> CBM Hackers responses:
>
> Hm... The way I read the datasheet of the 6551, you need
> to check the status register whether a byte is waiting
> (Bit 3 set) and if yes, grab the byte and store it into
> the buffer. That BEQ doesn't really make sense in this
> context.

It's been a while since I looked at this, but I believe the
BEQ is there to bypass the code that checks if the byte is
Xon or Xoff.

> If I remember well I have used the serial port as tty
> terminal in the past and it was working fine (although
> probably that does not use a 0x0 byte). Also at the
> university a guy has written a SLIP protocol software
> and could get IP packets. He has created telnet, ftp and
> it was working. It was the subject of his thesis and he
> has graduated.

I don't know if normal traffic would encounter null bytes,
but I would think file transfers might. In any case, it's
possible to avoid any problem if your software takes over
the beginning of the IRQ routine, duplicates it up to this
point, makes the fix, then jumps back into ROM. So the fact
that all his stuff worked doesn't mean the error isn't
there. He may have used his own code.

But I would just say that as far as I can tell the value
$fd00 occurs only twice in the entire rom, once to read from
that location, and once to write to it. It also seems
pretty clear that if it takes the BEQ, it then loads in the
value from $07D5 and writes it into the input buffer. If it
never writes a null into $07d5, there's no way a received

null will ever get into that buffer.

> I read your note that the receive routine will read the
> READ register of the 6551 ($fd00) and then go elsewhere
> of the value is 0, storing it at $0fd5 (though you also
> say $07d5, which confused me, maybe a typo?) if <>0.

Yes. A typo. Sorry. It's $07d5.

> The text, though, states that the routine will receive a
> null byte, not store it, but then when a non-null byte
> is received, it will store the null in the place the
> non-null byte was supposed to be stored. That would seem
> to be a huge issue, and I'm not aware anyone sees such
> behavior.

No. $07d5 is the temp storage location for the incoming byte. A non-null byte is first stored there, then later retrieved and stored into the buffer. A null byte is NOT stored in $07d5, but the value in $07d5 is retrieved anyway. The result would be that a null byte produces a duplicate of whatever the last non-null byte was.

> Gerrit's comment above is noting that the BEQ doesn't
> make any sense, as by the time the routine reads data
> from the READ register, it should always have previously
> checked the data available flag. If set, the data in
> the read register should be stored regardless, and no
> conditional should be performed.

I agree, except for the Xon/Xoff check. I'll have to double check that, but my memory is that it compares the received byte to zero-page locations that contain the values, if any, being used for Xon and Xoff. If Xon/Xoff is NOT being used, then those zero-page values are probably zeros, and in that case for a null byte you need to skip over the test because otherwise you would get a false match. I think that's why the BEQ is there.

George

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Tue, 16 Nov 2021 04:22:19 GMT
View Forum Message <> Reply to Message

Steven Combs sent me a copy of the Term-80 terminal program, the Plus/4 version, which works well on the Plus/4. I have nothing to test it on, but

was able to examine the .PRG file, and found that the author uses his own
IRQ code for handling the 6551. So that's a case where the rom error would
never come into play because the author uses his own replacement code
instead of the rom code. He does that by hijacking the IRQ ram vector at
$0314.

I've asked Steven to send me Plus/4 terminal software that doesn't work so
well. Maybe we can find one that doesn't bring in new ACIA handler code,
and we can see if any errors match the expected doubling of non-null
characters that the rom error should produce. Then if he can arrange to
flash a new eprom containg the fix, we can see if performance is improved.

Here's Steven's video on the Plus/4 6551:

https://www.youtube.com/watch?v=daoAllAv9qo

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by Jim Brain on Fri, 19 Nov 2021 16:30:50 GMT
View Forum Message <> Reply to Message

On 11/15/2021 9:50 AM, George wrote:
> Jim Brain says...
>
>> CBM Hackers responses:
>
>> Hm... The way I read the datasheet of the 6551, you need
>> to check the status register whether a byte is waiting
>> (Bit 3 set) and if yes, grab the byte and store it into
>> the buffer. That BEQ doesn't really make sense in this
>> context.
>
> It's been a while since I looked at this, but I believe the
> BEQ is there to bypass the code that checks if the byte is
> Xon or Xoff.

It might make sense to write up a bit more of the disassembly with your
notes to create clarity.
>
> I don't know if normal traffic would encounter null bytes,
> but I would think file transfers might. In any case, it's
> possible to avoid any problem if your software takes over
> the beginning of the IRQ routine, duplicates it up to this
> point, makes the fix, then jumps back into ROM. So the fact
> that all his stuff worked doesn't mean the error isn't
> there. He may have used his own code.The original post may have confused some folks, but I do think he was
aware we were discussing the stock routines, so I don't think he was

referring to home-spun code.
>
>
>
>> The text, though, states that the routine will receive a
>> null byte, not store it, but then when a non-null byte
>> is received, it will store the null in the place the
>> non-null byte was supposed to be stored. That would seem
>> to be a huge issue, and I'm not aware anyone sees such
>> behavior.
>
> No. $07d5 is the temp storage location for the incoming
> byte. A non-null byte is first stored there, then later
> retrieved and stored into the buffer. A null byte is NOT
> stored in $07d5, but the value in $07d5 is retrieved anyway.
> The result would be that a null byte produces a duplicate of
> whatever the last non-null byte was.
Ah, that clears things up for me. So, if $34 $00 were the data items,
the data delivered to the +4 app would be $34 $34
>
>
> I agree, except for the Xon/Xoff check. I'll have to double
> check that, but my memory is that it compares the received
> byte to zero-page locations that contain the values, if any,
> being used for Xon and Xoff. If Xon/Xoff is NOT being used,
> then those zero-page values are probably zeros, and in that
> case for a null byte you need to skip over the test because
> otherwise you would get a false match. I think that's why the
> BEQ is there.
Ah, understood.
>
> George
>

--
Jim Brain, brain@jbrain.com
RETRO Innovations: Contemporary Gear for Classic Systems
www.go4retro.com

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Fri, 19 Nov 2021 23:29:16 GMT
View Forum Message <> Reply to Message

Jim Brain says...

 > It might make sense to write up a bit more of the

> disassembly with your notes to create clarity.

See below.

> Ah, that clears things up for me. So, if $34 $00 were
> the data items, the data delivered to the +4 app would
> be $34 $34

Yes, that's right. I think regular BBS traffic probably
wouldn't have any nulls, so for that, this problem wouldn't
make any difference. But I think file transfers might have
lots of nulls, including any two-byte block numbers, or
two-byte checksums. And of course any program with an ML
section would have nulls all over the place (LDA #$00, STA
$FD00, and such).

Below is the relevant section of somebody's kernel source
code, and on the right side the more raw version of the same
thing showing the actual hex values.

The error results from an attempt to work software flow
control (Xon/Xoff) into the ACIA IRQ servicing routines. If
flow control is enabled, the values normally used for Xon
and Xoff ($11 and $13 respectively) will be stored at
locations $FC and $FD. If flow control is disabled, those
locations will contain nulls.

If flow control is disabled, a received null byte must not
be compared to $FC or $FD because a false match would be
detected, and we would be halting and resuming transmission
for no reason. So the code branches around all the Xon/Xoff
stuff if a null is received. But it fails to save the null
in aintmp ($07D5) before branching. So later, when the
value in aintmp is retrieved and added to the input queue,
another copy of the last non-null byte received is what will
go into the queue.

The solution is to reverse the two instructions, so the null
is saved into aintmp, then the BEQ is performed. The Zero
flag will not be modified by the STA instruction, so the
branch will work correctly.

```
ain
     lda  astat      ; acia status reg prev. saved    LEA95   LDA   $07D4
     and  #$8         ; bit 3 set if char recd.            AND   #$08
     beq  rxfull     ; no char has been received            BEQ   LEAF0
     lda  astat      ; got one...reset stat bit             LDA   $07D4
     and  #$f7                                          AND   #$F7
     sta  astat                                 STA   $07D4
     lda  acia       ; read byte                    LDA   $FD00
     beq  notacc      ; if null, skip xon/xoff              BEQ   LEAC2

; it's a null, don't let thru for x-disable

     sta  aintmp      ; save char [unless it was a null]       STA   $07D5
     cmp  xon         ; is it a ~q                     CMP   $FC
     bne  trycs       ; nope                          BNE   LEAB7

; got a ~q

     lda  #0                                    LDA   #$00
     sta  alstop      ; tell local xmit to go               STA   $07D6
     beq  rxfull      ; !bra, what character?               BEQ   LEAF0

trycs
     cmp  xoff        ; is it a ~s                  LEAB7   CMP   $FD
     bne  notacc      ; nope                            BNE   LEAC2

; got a ~s

     lda  #$ff                                  LDA   #$FF
     sta  alstop      ; tell local xmit to stop              STA   $07D6
     bne  rxfull      ; !bra, i didn't see that...           BNE   LEAF0

notacc
     lda  inqcnt                              LEAC2   LDA   $07D3
     cmp  #inpqln-1   ; is queue full                     CMP   #$3F
     beq  rxfull      ; yep                           BEQ   LEAF0
     cmp  #hiwatr     ; high water mark                    CMP   #$38
     bne  nohw        ; nope                            BNE   LEADC

; hit high water mark, tell sender to stop

     lda  xoff        ; x-sw is off                    LDA   $FD
     beq  nohw                                    BEQ   LEADC
     sta  soutq       ; ~s                            STA   $07CF
     lda  #$ff                                  LDA   #$FF
     sta  soutfg      ; flag it present                   STA   $07D0
     sta  arstop      ; flag remote stopped                  STA   $07D7

nohw
                 ; not full, insert char
     ldx  inqfpt      ; do: inqfpt <- inqfpt+1 mod 64  LEADC   LDX   $07D1
     inx                                       INX
     txa                                       TXA
     and  #$3f                                   AND   #$3F
     sta  inqfpt                                 STA   $07D1
     tax                                       TAX
     lda  aintmp      ; get char to insert              LDA   $07D5
     sta  inpque,x    ; insert it                    STA $03F7,x
     inc  inqcnt      ; another drop in the bucket        INC   $07D3
rxfull               ; error exit
     rts             ; all ok                   LEAF0   RTS
```

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Thu, 25 Nov 2021 23:57:04 GMT

View Forum Message <> Reply to Message

Just to bring this to a conclusion, I've written alternative IRQ servicing
code that bypasses the "BEQ" error in rom, and a BASIC program that
installs the new code into the tape buffer. It's 120 bytes of ML that does
the ACIA processing and then jumps back into ROM for the rest of the IRQ
routine. The BASIC program and the source code for the ML portion are
shown below. The actual .PRG file can be downloaded from my Github CBM
repo:

https://github.com/gbhug5a/My_CBM_stuff

Since I no longer have a +4, I have no way to test the new code.

```
10 rem this code bypasses the "beq" bug
12 rem in the +4's irq acia received-
14 rem byte routine, and eliminates the
16 rem xon/xoff software flow control
18 rem code in both xmit and receive.
20 rem the irq service routine vector
22 rem ($0314) normally points to $ce0e.
24 rem this code duplicates that code up
26 rem to the acia, fixes that, then
28 rem jumps back into rom.  the code is
30 rem poked into the cassette buffer at
32 rem $0333, but can be placed anywhere
34 rem visible in ram when kernel and
36 rem basic roms are banked in.  sys
38 rem to the first byte to take over
40 rem $0314. no need to re-assemble for
42 rem a different location. the code
44 rem detects where it has been placed.
46 rem sys (first byte + 26) to restore
48 rem the the $0314 vector to $ce0e.
100 cb = 819
110 for i = 0 to 119
120 read a
```

```
130 poke cb+i,a
140 next
150 sys cb
160 v= (peek(789)*256) + peek(788)
170 print "irq ram vector now";v
180 data 120,32,85,252,186,202,189,0
190 data 1,24,105,36,141,20,3,232
200 data 189,0,1,105,0,141,21,3
210 data 88,96,120,169,14,141,20,3
220 data 169,206,141,21,3,88,96,173
230 data 9,255,41,2,240,3,32,96
240 data 206,44,216,7,16,63,173,1
250 data 253,141,212,7,16,55,173,212
260 data 7,41,8,240,24,173,212,7
270 data 41,247,141,212,7,173,0,253
280 data 141,213,7,173,211,7,201,63
290 data 240,3,32,220,234,173,212,7
300 data 41,16,240,17,173,16,253,41
310 data 2,240,10,162,0,44,206,7
320 data 16,3,32,131,234,76,43,206
```

```
..6502

;code to bypass error in +4 acia irq receive-byte routine,
;and eliminate xon/xoff software flow control for
;transmit and receive.

..org $0333          ;object code can be moved anywhere
                     ;  without reassembly
                     ;sys entry to set irq vector to newirq
                     ;sys restore (entry + 26) to restore
       ;  default vector

entry:

   sei
   jsr   $fc55       ;just rts there
   tsx               ;pc now in stack
   dex
   lda   $0100,x
   clc
   adc   #(newirq - entry - 3) ;point to newirq
   sta   $0314
   inx
   lda   $0100,x
   adc   #0
   sta   $0315       ;irq vector now newirq
   cli
```

```
  rts              ;return from sys

restore:            ;restore = entry + 26

  sei              ;restore vector to $ce0e
  lda   #$0e
  sta   $0314
  lda   #$ce
  sta   $0315
  cli
  rts

newirq:             ;$035a (858) if entry = $0333

  lda   $ff09       ;not related to acia
  and   #$02
  beq   checkacia
  jsr   $ce60

checkacia:

  bit   $07d8       ;acia present?
  bpl   backtorom
  lda   $fd01       ;read status reg
  sta   $07d4       ;save status reg
  bpl   backtorom   ;bit 7 set if acia triggered irq

receive:

  lda   $07d4       ;new byte received?
  and   #$08
  beq   transmit
  lda   $07d4
  and   #$f7
  sta   $07d4
  lda   $fd00       ;new byte
  sta   $07d5

  lda   $07d3       ;number of bytes in queue
  cmp   #$3f
  beq   transmit    ;discard byte if full

  jsr   $eadc       ;add new byte to input queue

transmit:

  lda   $07d4
  and   #$10        ;transmit buffer empty?
  beq   backtorom
  lda   $fd10       ;pin k of user port = cts
  and   #$02
  beq   backtorom   ;modem says don't send
  ldx   #$00
  bit   $07ce       ;anything to send?
  bpl   backtorom
  jsr   $ea83       ;send it

backtorom:

  jmp   $ce2b       ;acia done, continue rest of irq
```

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by Jim Brain on Sat, 27 Nov 2021 06:34:10 GMT

View Forum Message <> Reply to Message

On 11/25/2021 5:57 PM, George wrote:
> Just to bring this to a conclusion, I've written alternative IRQ servicing
> code that bypasses the "BEQ" error in rom, and a BASIC program that
> installs the new code into the tape buffer. It's 120 bytes of ML that does
> the ACIA processing and then jumps back into ROM for the rest of the IRQ
> routine. The BASIC program and the source code for the ML portion are
> shown below. The actual .PRG file can be downloaded from my Github CBM
> repo:
>
> https://github.com/gbhug5a/My_CBM_stuff
>
> Since I no longer have a +4, I have no way to test the new code.

Do you have the VICE emulator? It supports the +4 and the built in
6551, and can tie that 6551 ACIA to a real or virtual serial port. You
could test it there if desired.

Jim

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Sun, 28 Nov 2021 21:45:15 GMT

View Forum Message <> Reply to Message

Jim Brain says...

> Do you have the VICE emulator? It supports the +4 and
> the built in 6551, and can tie that 6551 ACIA to a real
> or virtual serial port. You could test it there if

> desired.

I haven't been able to get the Plus/4 ACIA emulation to work
in VICE. The rom emulation knows about the ACIA, and will
let me open a channel to it, and sets the command and
control registers, but it doesn't actually do anything, such
as generating interrupts. I tried the +acia command line
switch, but then it doesn't even let me open the channel.
It errors out with Device Not Present.

If you know of a way to get 6551 emulation to function,
please give me the exact sequence of what I need to do.

Subject: Re: Plus 4 rom error - is there any place to report it?
Posted by George on Sun, 28 Nov 2021 23:51:27 GMT

View Forum Message <> Reply to Message

If anyone has a Plus/4 and has nothing better to do, it would be helpful if
you could run the programs listed below and report the results. The
programs report the total number of ACIA interrupts which occur while
transmitting 256 bytes of data continuously at 2400 baud. They should be
run with nothing plugged into the User Port - no modem or anything.

My suspicion is that the single-byte transmit buffer of the Plus/4 results
in back-to-back double interrupts being generated for each byte
transmitted, which can cause problems at high speed. If that's the case,
the first program will report about 512 interrupts, and the second about
256. But there may be no difference. Anyway, I just need to know.

irqtest1

```
10 rem  irq test1 to count interrupts
20 cb = 819
30 for i = 0 to 161
40 read a
50 poke cb+i,a
60 next
70 sys cb
75 open 2,2,0,chr$(26)+chr$(7)
80 sys 940
85 close 2
90 sys 845
95 print peek(1009)*256 + peek(1008)
100 data 120,32,85,252,186,202,189,0
110 data 1,24,105,36,141,20,3,232
120 data 189,0,1,105,0,141,21,3
```

```
130 data 88,96,120,169,14,141,20,3
140 data 169,206,141,21,3,88,96,173
150 data 9,255,41,2,240,3,32,96
160 data 206,44,216,7,16,64,173,1
170 data 253,141,212,7,16,56,238,240
180 data 3,208,3,238,241,3,173,212
190 data 7,41,8,240,24,173,212,7
200 data 41,247,141,212,7,173,0,253
210 data 141,213,7,173,211,7,201,63
220 data 240,3,32,220,234,173,212,7
230 data 41,16,240,10,44,206,7,16
240 data 5,162,0,32,131,234,76,43
250 data 206,120,169,0,141,240,3,141
260 data 241,3,141,242,3,88,44,206
270 data 7,48,251,169,85,141,205,7
280 data 56,110,206,7,206,242,3,208
290 data 237,173,2,253,73,12,141,2
300 data 253,96
```

irqtest2

```
10 rem  irq test2 to count interrupts
20 cb = 819
30 for i = 0 to 173
40 read a
50 poke cb+i,a
60 next
70 sys cb
75 open 2,2,0,chr$(26)+chr$(7)
80 sys 952
85 close 2
90 sys 845
95 print peek(1009)*256 + peek(1008)
100 data 120,32,85,252,186,202,189,0
110 data 1,24,105,36,141,20,3,232
120 data 189,0,1,105,0,141,21,3
130 data 88,96,120,169,14,141,20,3
140 data 169,206,141,21,3,88,96,173
150 data 9,255,41,2,240,3,32,96
160 data 206,44,216,7,16,76,173,1
170 data 253,141,212,7,16,68,238,240
180 data 3,208,3,238,241,3,173,212
190 data 7,41,8,240,24,173,212,7
200 data 41,247,141,212,7,173,0,253
210 data 141,213,7,173,211,7,201,63
220 data 240,3,32,220,234,173,212,7
230 data 41,16,240,22,44,206,7,16
```

```
240 data 17,172,2,253,152,73,12,141
250 data 2,253,162,0,32,131,234,140
260 data 2,253,76,43,206,120,169,0
270 data 141,240,3,141,241,3,141,242
280 data 3,88,44,206,7,48,251,169
290 data 85,141,205,7,56,110,206,7
300 data 206,242,3,208,237,173,2,253
310 data 73,12,141,2,253,96
```