Subject: Generating a timed IRQ with a 6530 (6532)?
Posted by Paul Förster on Sat, 07 Jun 2014 13:36:38 GMT

Hi,

I'm (trying to) write a monitor program for the KIM-1 or Micro-KIM. I would like to implement a "W"alk command which interrupts after each 6502 instruction's execution, i.e. a single step mode. Yes, I know that the KIM-1 has an SST switch but I'm in serial terminal mode and I don't want to use that to keep my fingers on the keyboard rather than playing around with very sensitive mini keys. ;-)

I searched around the net for ideas and found the BRK method which simply replaces the byte after the instruction to be executed with a BRK, needing proper handling of branch and jump instructions of course. This method has the disadvantage that it's relatively expensive code-wise because it needs to know the length of any instruction as well as handle branches and jumps manually. I don't like it for these reasons.

Then I found another idea which really triggered my interest. If an IRQ occurs *during* the execution of an instruction then the instruction's execution is finished and then, after that, the IRQ is processed. Hence my idea: Generate an IRQ which should occur during a specified number of cycles with one of the two 6530 (or 6532 on the Micro-KIM respectively) and with that, trigger my monitor re-entry through the IRQ. So far, so good. It *should* work in theory, at least I think or hope so.

Now, how would I do this? I set the IRQ vector to an entry point in my monitor, which works, well at least as tested with the BRK command. I don't know about how to trigger other interrupt sources, though.

What I need would be something like this:

```
walkcmd  ...       ; setup code for 6530 IRQ timer for IRQ generation
         jmp (pcl) ; jmp to instruction to execute
```

The interrupt should occur somewhere during the execution of the instruction the JMP (PCL) jumps to, i.e. very shortly after the jump itself has finished. How can I achieve this?

Any idea would be appreciated. Thanks very much in advance.
--
cul8er

Paul

paul.foerster@gmx.net

---

Subject: Re: Generating a timed IRQ with a 6530 (6532)?
Posted by Anton Treuenfels on Sat, 07 Jun 2014 17:49:50 GMT
View Forum Message <> Reply to Message

"Paul Förster" <paul.foerster@gmx.net> wrote in message
news:bvgir7Fp350U1@mid.individual.net...
> Hi,
>
> I'm (trying to) write a monitor program for the KIM-1 or Micro-KIM. I
> would like to implement a "W"alk command which interrupts after each 6502
> instruction's execution, i.e. a single step mode. Yes, I know that the
> KIM-1 has an SST switch but I'm in serial terminal mode and I don't want
> to use that to keep my fingers on the keyboard rather than playing around
> with very sensitive mini keys. ;-)

http://www.sleepingelephant.com/ipw-web/bulletin/bb/viewtopi c.php?f=11&t=6511

The relevant portion being

==============:

[..] KIM-1 did a semi hardware trick. On the 6502 there is a signal called
SYNC. That will go HIGH whenever the CPU is fetching an opcode. KIM-1 tied
that to the NMI signal (through a switch and gate). Every opcode fetch would
then generate an NMI interrupt and you could display PC, Registers, etc.

===============

- Anton Treuenfels

---

Subject: Re: Generating a timed IRQ with a 6530 (6532)?
Posted by Paul Förster on Sat, 07 Jun 2014 19:03:58 GMT
View Forum Message <> Reply to Message

Hi Anton,

On 2014-06-07 17:49:50 +0000, Anton Treuenfels said:
> http://www.sleepingelephant.com/ipw-web/bulletin/bb/viewtopi c.php?f=11&t=6511
> The relevant portion being
> ==============:
> [..] KIM-1 did a semi hardware trick. On the 6502 there is a signal
> called SYNC. That will go HIGH whenever the CPU is fetching an opcode.
> KIM-1 tied that to the NMI signal (through a switch and gate). Every

> opcode fetch would then generate an NMI interrupt and you could display
> PC, Registers, etc.
> ===============

thanks for the pointer. Well, "through a switch"... that was what I was
trying to avoid. I'd rather say the relevant portion is:

"As I recall hesmon uses the NMI and a timer to achieve this.
When you issue the (W)alk or (Q)uick Trace command it sets up the NMI
routine for the job and sets the VIA#1 timer to timeout after just a
few clock ticks. Enough to do Start the Walk/Quick Trace.
The CPU then only have time to execute the next op-code before the
timer times out and a NMI is issued. Control is then handed over to the
NMI routine. This routine checks for breakpoints and either waits for a
key press or continuous. After the house work the VIA timer is again
set and return from NMI interrupt is executed. The CPU now executes
another op-code, timer times out and another NMI is fired."

The question for me is, how can I setup the 6530's timer to trigger an
IRQ or NMI. I have no experience with such stuff on the 6530 chip. Is
there sample code somewhere? I didn't find any.
--
cul8er

Paul
paul.foerster@gmx.net

---

## Subject: Re: Generating a timed IRQ with a 6530 (6532)?
## Posted by Anton Treuenfels on Sun, 08 Jun 2014 21:08:10 GMT

View Forum Message <> Reply to Message

"Paul Förster" <paul.foerster@gmx.net> wrote in message
news:bvh611Ft1c4U1@mid.individual.net...
> Hi Anton,
>
> On 2014-06-07 17:49:50 +0000, Anton Treuenfels said:
>> http://www.sleepingelephant.com/ipw-web/bulletin/bb/viewtopi c.php?f=11&t=6511
>> The relevant portion being
>> ==============:
>> [..] KIM-1 did a semi hardware trick. On the 6502 there is a signal
>> called SYNC. That will go HIGH whenever the CPU is fetching an opcode.
>> KIM-1 tied that to the NMI signal (through a switch and gate). Every
>> opcode fetch would then generate an NMI interrupt and you could display
>> PC, Registers, etc.
>> ===============

I have no experience with the KIM-1, but as I read this it means you flip

the switch, and then until you flip it again you get an NMI on every
instruction fetch. Ie., you do not flip the switch for every instruction.
Which was what I understood you were trying to avoid. But if you absolutely
must do it in software...

> thanks for the pointer. Well, "through a switch"... that was what I was
> trying to avoid. I'd rather say the relevant portion is:
>
> "As I recall hesmon uses the NMI and a timer to achieve this.
> When you issue the (W)alk or (Q)uick Trace command it sets up the NMI
> routine for the job and sets the VIA#1 timer to timeout after just a few
> clock ticks. Enough to do Start the Walk/Quick Trace.
> The CPU then only have time to execute the next op-code before the timer
> times out and a NMI is issued. Control is then handed over to the NMI
> routine. This routine checks for breakpoints and either waits for a key
> press or continuous. After the house work the VIA timer is again set and
> return from NMI interrupt is executed. The CPU now executes another
> op-code, timer times out and another NMI is fired."
>
> The question for me is, how can I setup the 6530's timer to trigger an IRQ
> or NMI. I have no experience with such stuff on the 6530 chip. Is there
> sample code somewhere? I didn't find any.
> --

Dunno, never programmed the 6530, only the 6526. The "trick" that makes it
work, I suspect, is that the minimum instruction cycle time is two. So
HESMon probably set the C64's 6526 (or the VIC-20's 6522) to issue a
interrupt after only one cycle (also likely in one-shot mode rather than
continuous). Might have had to modify the actual interrupt time to account
for any instructions always executed after actually starting the timer, but
that's the general idea.

You can find a data sheet for the 6530 at:

http://6502.org/documents/datasheets/mos/

From that you should be able to figure out how to program it.

Of course if you know how the hardware of your KIM is connected, you'll be
able to check if there already is that hardware switch thing in place. Or if
the 6530 can actually interrupt the 6502 (ie., is the output of the 6530
connected to the 6502?). Either way the majority of your service routine is
likely to be similar or identical. It's just the "getting in and getting
out" parts the will likely differ.

- Anton Treuenfels

Subject: Re: Generating a timed IRQ with a 6530 (6532)?
Posted by Paul Förster on Sun, 08 Jun 2014 21:49:36 GMT
View Forum Message <> Reply to Message

Hi Anton,

On 2014-06-08 21:08:10 +0000, Anton Treuenfels said:
> I have no experience with the KIM-1, but as I read this it means you
> flip the switch, and then until you flip it again you get an NMI on
> every instruction fetch. Ie., you do not flip the switch for every
> instruction. Which was what I understood you were trying to avoid. But
> if you absolutely must do it in software...

the KIM-1 has a single-step switch. One has to set the IRQ and NMI
vectors to $1c00 for it to work. Once the setting is in place, one can
flip the switch to single step and then execute any non-ROM program
instruction by instruction. After each instruction the ROM monitor is
entered again and one can examine the machine status and then can
continue execution of the next instruction. But the SST switch has to
be turned on for this to work. My intentions was to do that via the
6530 timer, which can do a one shot.

> Dunno, never programmed the 6530, only the 6526. The "trick" that makes
> it work, I suspect, is that the minimum instruction cycle time is two.
> So HESMon probably set the C64's 6526 (or the VIC-20's 6522) to issue a
> interrupt after only one cycle (also likely in one-shot mode rather
> than continuous). Might have had to modify the actual interrupt time to
> account for any instructions always executed after actually starting
> the timer, but that's the general idea.

yes, the general idea is pretty clear.

> You can find a data sheet for the 6530 at:
> http://6502.org/documents/datasheets/mos/
> From that you should be able to figure out how to program it.

there are tons of copies everywhere. What I don't get is, how to
program the one shot timer.

> Of course if you know how the hardware of your KIM is connected, you'll
> be able to check if there already is that hardware switch thing in
> place. Or if the 6530 can actually interrupt the 6502 (ie., is the
> output of the 6530 connected to the 6502?). Either way the majority of
> your service routine is likely to be similar or identical. It's just
> the "getting in and getting out" parts the will likely differ.

the service routine should be identical. That's not the problem for me.
My problem is to get the chip fire the interrupt at all. Once I've
managed to do that, I can adjust the number of cycles which it has to

wait before doing so.
--
cul8er

Paul
paul.foerster@gmx.net

---

Subject: Re: Generating a timed IRQ with a 6530 (6532)?
Posted by Anton Treuenfels on Mon, 09 Jun 2014 04:13:23 GMT
View Forum Message <> Reply to Message

"Paul Förster" <paul.foerster@gmx.net> wrote in message
news:bvk43iFh3j4U1@mid.individual.net...
> the service routine should be identical. That's not the problem for me. My
> problem is to get the chip fire the interrupt at all. Once I've managed to
> do that, I can adjust the number of cycles which it has to wait before
> doing so.

From the data sheet it doesn't look too different from the 6522 or the 6526.
If you can find and puzzle out the logic for an IRQ servicing routine for
one of those the principles should transfer to the 6530. The VIC-20 and C64
Kernel ROMS use timed IRQs for RS-232 servicing, so there's one place to
look.

The 6530 has an A3 line that enables or disables the IRQ and an 8-bit port
used to set the timing of the IRQ. Also a setting for how many clock cycles
cause the timer to decrement one unit. Setting all those is simply a matter
of writing to the locations where they appear in the KIM's memory map.

- Anton Treuenfels

---

Subject: Re: Generating a timed IRQ with a 6530 (6532)?
Posted by Paul Förster on Mon, 09 Jun 2014 07:30:01 GMT
View Forum Message <> Reply to Message

Hi Anton,

On 2014-06-09 04:13:23 +0000, Anton Treuenfels said:
> From the data sheet it doesn't look too different from the 6522 or the
> 6526. If you can find and puzzle out the logic for an IRQ servicing
> routine for one of those the principles should transfer to the 6530.
> The VIC-20 and C64 Kernel ROMS use timed IRQs for RS-232 servicing, so
> there's one place to look.

LOL. Almost the whole 2 KB of the KIM-1 are about serial transmission. :-P

> The 6530 has an A3 line that enables or disables the IRQ and an 8-bit
> port used to set the timing of the IRQ. Also a setting for how many
> clock cycles cause the timer to decrement one unit. Setting all those
> is simply a matter of writing to the locations where they appear in the
> KIM's memory map.

yes, I know all that. I'd do something like the following:

```
lda #cyc
sta clk1t
lda #cyc8
sta clk8t
lda #cyc64
sta clk64t
lda #cyc1k
sta clkkt
```

effectively having cyc8, cyc64 and cyc1k probably 0 as I only need a few cycles. That should set the count-down cycle counter.

Yet, how and where can I set this ominous A3 line. I've read the chip description but (this is generally my main problem when reading any data sheet) it does not clearly map to the machine in my mind. I know that data register PAD is mapped to $1700, data direction register PADD to $1701 and ditto for PBD and PBDD to $1702 and $1703 respectively. Interestingly and confusingly enough, the KIM-1 ROM listing has those at different addresses:

```
 24          SAD    =   $1740    6530 A DATA
 25          PADD   =   $1741    6530 A DATA DIRECTION
 26          SBD    =   $1742    6530 B DATA
 27          PBDD   =   $1743    6530 B DATA DIRECTION
 28          CLK1T  =   $1744    DIV BY 1 TIME
 29          CLK8T  =   $1745    DIV BY 8 TIME
 30          CLK64T =   $1746    DIV BY 64 TIME
 31          CLKKT  =   $1747    DIV BY 1024 TIME
```

As the code in the ROM listing is the actual running code, I'd rather take that as an authority.
--
cul8er

Paul
paul.foerster@gmx.net