Subject: Linux on KISS-68030 single board computer
Posted by will on Sun, 21 Feb 2016 22:30:03 GMT
View Forum Message <> Reply to Message

Hello everyone!

I am pleased to announce my first release of Linux for the KISS-68030!

I've made a slightly rambling video to show the board and demonstrate it working:
https://www.youtube.com/watch?v=MYpMOBaohbw

Not sure how the group feels about the video format - I think it's a good way to properly demonstrate a system which others may not have access to. Feedback is welcomed -- and I'd love to see videos from other SBC and S-100 builders!

Current status:

ROM supports loading COFF, ELF executables from FAT filesystem Linux has been ported to the machine and I have written drivers for all of the hardware System appears to be very stable System is self-hosting; the ROM and the Linux kernel can both be built on the machine itself.

Required hardware:

Power supply (I used my bench power supply set to 5V 2A) ECB backplane KISS-68030 CPU board MF/PIC I/O board (you must first apply the 2015-09-22 update to connect IDE IRQ to NS32202 IRQ9) 8GB CompactFlash card and IDE adapter (possibly a regular hard disk will work also)

I have uploaded a ZIP file (331MB) which contains:

Boot ROM image Patch against Linux 4.4.1 source code README file with further instructions CompactFlash card disk image: This includes a boot filesystem and a current Debian Linux root filesystem with a bunch of standard tools. This has gcc and the other tools required to rebuild the kernel and ROM from the source code (also included).

I think at this stage the Linux support for this board is complete, pending hardware additions! My next objective is to connect the machine to a TCP/IP network. I think the simplest way to achieve this will be to add another UART for a PPP serial link. Initially I will use a second MF/PIC board with some modifications, essentially omitting the NS32202 and DS1302 and re-routing the interrupts. I am waiting for the PCB and components to arrive so I can start work on that. John is also working on a quad-UART board design which will be a better solution when it becomes available.

Copy of the README file follows. Share and enjoy!

Debian Linux for KISS-68030

To make this work:

- configure your MF-PIC at a base address of 0x40 (which is the standard setting).

- fit as much RAM as possible. I suspect 16MB is the minimum required to boot to multiuser.

- fast CPU recommended.

- get an 8GB or larger Compact Flash card, and write the disk image to it:
  bunzip -dc 2016-02-20-kiss68030-hddimage.dd.bz2 | dd bs=1M of=/dev/sdX
  where /dev/sdX is the name of your CF card (note: the whole card, not a partition on it, ie /dev/sde instead of /dev/sde1). The Compact Flash card contains:
  - First partition (128MB): A FAT format filesystem, used by the boot ROM.
  - Second partition (about 7GB): An ext4 format filesystem, used by Linux.
  - Third partition (1GB): Swap space for Linux. Try to avoid using this!

- Connect the CF card to the IDE port on the MF-PIC.

- write "kissbios.rom" to your ROM chip -- I have padded the file to 512KB to make this easier, however only the first 46KB is actually used so you could use a smaller flash ROM chip if you need to (this is untested)

- set your terminal to 115200 baud, 8 bit data, no parity, 1 stop bit, and enable RTS/CTS hardware flow control.

- power on the system. On the first boot you will want to run Setup by pressing "s" immediately on power-on. Enable IDE Board Type 1 ("Parallel Port") at Base port address 44 hex. Then on rebooting the system it should run the ROM. This should detect the CF card, find "boot.cmd" on it, and boot it automatically after 1--2 seconds.

- Linux should boot and after about 10 minutes you should be able to log in with username "root", password "root".

- Useful tricks:
  rz, sz -- Z-Modem file transfer through console
  screen -- Run multiple sessions on one terminal (read the man page)
  flash030 -- Reprogram the flash ROM in-system
  nice -- Run a program with reduced CPU priority

- When you're finished, type "halt".

Source code and build environment:

- Source code for the Linux kernel, boot ROM and flash030 is all in the /root/ directory on the Linux filesystem. The filesystem image includes all the tools to build these on the KISS-68030 system itself, or you can cross-compile on a faster machine.

Notes on the ROM:

- It is only really useful with a CF card present.

- If a file named "boot.cmd" is present it will be used as a list of commands to automatically boot the system.

- If "boot.cmd" is not present the system will drop to the ROM command line rather than automatically booting.

- Pressing "x" or "s" quickly after reset will drop to the command line or enter setup, respectively. Note that any other key will skip the delay and automatically boot immediately.

- At the ROM command line:

  - If you have multiple disks, 0:, 1:, 2:, 3: etc will switch between them

  - "cd" will change directory

  - "ls" or "dir" will list the directory.

  - typing a file name will try to load it as a program:

    - COFF or ELF executables are supported. A single argument of "u" or "s" will cause the program to run in user or supervisor mode, respectively:
      "myprog.elf u"

    - Linux kernels are supported. Arguments are passed to the kernel verbatim, with the exception of "initrd=<filename>" which can be given to load an initrd image into memory for the kernel:
      "vmlinux.elf initrd=initrd.img console=ttyS0,115200n8r init=/bin/bash"

    - Any other arbitrary file can be loaded by giving the load-address as address (in hex):
      "myprog.bin 1000"

  - "execute <addr> [u|s]" will jump to a given address in user or supervisor mode.

  - "writemem <addr> <byte...>" will write to a given address in memory. You

can use "wm" as a shorthand.

- "dump <addr> <count>" will dump the contents of the specified region of memory in hex and ASCII. You can use "dm" as a shorthand.

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by Andrew B on Mon, 22 Feb 2016 04:01:16 GMT
View Forum Message <> Reply to Message

This is awesome!

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by will on Mon, 22 Feb 2016 12:32:16 GMT
View Forum Message <> Reply to Message

I've also prepared a ZIP file with a disk image for a 4GB CF card.

... if that is still too large try this disk image which will fit onto "four marketing-gigabyte" CF cards; you'll still need the other bits including the boot ROM from one of the ZIP files.

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by Andrew B on Mon, 22 Feb 2016 15:57:23 GMT
View Forum Message <> Reply to Message

Will, how difficult was it to write the custom patches to make this all work? I've looked at 68xxx-series Linux stuff a few times and it has been really hard to find good information - the m68k website is from 2001 and talks about kernel 2.x, the Debian wiki sites are way out of date, etc. I wasn't even aware that anyone was working on new m68k kernels.....

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by w9gb on Mon, 22 Feb 2016 18:02:24 GMT
View Forum Message <> Reply to Message

Will -

I watched the YouTube video.
You noted that the "boot process" for the Linux kernel takes about 10 minutes.
Obviously, the 68030 processor is not a modern multi-core processor.

Is this Linux image, just a basic OS core
with hardware drivers for KISS-68030 and companion MF/PIC I/O-board???

greg

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by will on Mon, 22 Feb 2016 18:16:54 GMT
View Forum Message <> Reply to Message

Hi Andrew

The linux-m68k mailing list is still active, the Linux "MAINTAINERS" file suggests that m68k is being maintained by Geert Uytterhoeven which does appear to be the case. The debian-68k mailing list is also still active. Debian m68k packages are being built (mostly, it seems, using the ARAnyM emulator) and are made available from debian-ports.org. So there is still some community interest.

I do agree a lot of the easily found information on the web is no longer useful!

To write my patches I simply dug into the code to figure out what was needed -- I found the entry vector (arch/m68k/kernel/head.S) and just followed along from there. Fortunately the 68K code is well designed and already supports a number of platforms with clean and logical separation so it was pretty obvious how to add another. To get the system booting on the processor it is really just a case of filling in the blanks with machine-specific code. The device drivers required a bit more work but nothing overly complex -- the KISS-68030 is pretty simple and clean.

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by will on Mon, 22 Feb 2016 18:24:23 GMT
View Forum Message <> Reply to Message

Hi Greg

To clarify -- to boot a full Debian system to a "login:" prompt (runlevel 2) takes about 10 minutes.

For most of my testing I have been booting the kernel and just running bash. To do this I boot with the argument "init=/bin/bash". This is very quick, about 20 seconds from hitting enter in the ROM to seeing the bash prompt appear.

The disk image I have supplied is a full Debian system - kernel which has full support for the KISS-68030 hardware, userspace which has a bunch of packages installed including python, perl, gcc, make, etc, and source code for the kernel, ROM, and my flash030 utility.

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by w9gb on Mon, 22 Feb 2016 19:12:34 GMT
View Forum Message <> Reply to Message

Will -
Perfect.
About 20 seconds to load kernel and then have bash prompt ...
that jarred my memory (Apple Mac II, NeXT Cube) from 25+ years ago (has it really been that long!!).
It is almost 30 years since the 68030 was introduced by Motorola

You also answered my follow-up question: You are loading the full Debian system (hence the 10-minutes).
My personal preference is pack light (years of airline travel will do that), so that can be improved by lightening the load.

Greg

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by will on Tue, 23 Feb 2016 23:22:49 GMT
View Forum Message <> Reply to Message

Quick update: John reports that his testing has determined 32MB is the minimum RAM to boot the system.

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by jcoffman on Wed, 24 Feb 2016 20:17:15 GMT
View Forum Message <> Reply to Message

Yes. I tried on a 16Mb system, the smallest memory size supported by the ECB M68030 board, but ran out of memory. My initial boot success was on a full 256Mb memory complement. With Linux up and running, "cat /proc/meminfo" indicates that most of that memory is unused.

Neither the limited 32Mb system, nor the full 256Mb system seems to do any swapping, at least as far as the utilities I have run.

The system seems to be CPU limited, 25Mhz in my case, not PPIDE limited in accessing the CF card "disk".

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by will on Wed, 24 Feb 2016 20:22:09 GMT
View Forum Message <> Reply to Message

Yes, the system is certainly CPU limited. I spent a bit of time optimising the disk I/O -- with my 32MHz CPU it runs at about 750KB/sec which is not bad. I made it go slightly faster (single digit %) by unrolling the inner loop but removed this for other reasons.

The transfer functions look like this, suggestions for improvements welcomed!

```
kiss68030_ide_data_input:
    moveal %sp@(4),%a0          /* void *buf */
    moveal %sp@(12),%a1         /* 8255 base address */
    movel %sp@(8),%d1           /* int dwords */

    /* save registers */
    movem.l %d2-%d3/%a2,-(%sp)

    lea %a1@(3),%a2             /* 8255 control register */
    moveq #13, %d2
    moveq #12, %d3

    /* skip to the end - to test for dwords=0 */
    jbra ide_input_loop

ide_input_nextword:
    /* read a DWORD */
    moveb %d2, %a2@             /* begin /RD pulse */
    movew %a1@, %d0             /* reads LSB then MSB in that order */
    moveb %d3, %a2@             /* end /RD pulse */
    swap %d0                    /* move to the top half of the word */
    moveb %d2, %a2@             /* begin /RD pulse */
    movew %a1@, %d0             /* reads LSB then MSB in that order */
    moveb %d3, %a2@             /* end /RD pulse */

    /* store to memory */
    movel %d0, %a0@+

    /* loop until done */
ide_input_loop:
    dbra %d1, ide_input_nextword

    /* restore registers, return */
    movem.l (%sp)+,%d2-%d3/%a2
    rts


kiss68030_ide_data_output:
    moveal %sp@(4),%a0          /* void *buf */
    moveal %sp@(12),%a1         /* 8255 base address */
    movel %sp@(8),%d1           /* int dwords */
```

```
    /* save registers */
    movem.l %d2-%d3/%a2,-(%sp)

    lea %a1@(3),%a2          /* 8255 control register */
    moveq #11, %d2
    moveq #10, %d3

    /* note that to give the drive time to latch the data we do
       housekeeping including swaps and branches while the /WR
       line is asserted -- so the first thing we do each loop is
       end the previous cycle's /WR pulse, which is a NOP on the
       entry to the first loop */

    /* skip to the end - to test for dwords=0 */
    jbra ide_output_loop
ide_output_nextword:
    /* load from memory */
    movel %a0@+, %d0
    /* write a DWORD */
    moveb %d3, %a2@          /* end /WR pulse -- nop on the first loop*/
    swap %d0                 /* top half first */
    movew %d0, %a1@          /* set up data lines */
    moveb %d2, %a2@          /* begin /WR pulse */
    swap %d0                 /* get the bottom half */
    moveb %d3, %a2@          /* end /WR pulse */
    movew %d0, %a1@          /* set up data lines */
    moveb %d2, %a2@          /* begin /WR pulse */
    /* loop until done */
ide_output_loop:
    dbra %d1, ide_output_nextword

    moveb %d3, %a2@          /* end the final /WR pulse */

    /* restore registers, return */
    movem.l (%sp)+,%d2-%d3/%a2
    rts
```

Subject: Re: Linux on N8VEM KISS-68030 single board computer
Posted by Andrew B on Sun, 28 Feb 2016 06:43:49 GMT
View Forum Message <> Reply to Message

Will, is the Debian userspace you used new enough to have systemd?

I bet there are a lot of opportunities to get that full userspace boot time down. Here is a presentation a guy at Ti gave about reducing userspace boot times from 120 seconds to 1 second

on a Ti development board - http://elinux.org/images/b/b3/Elce11_koen.pdf - a factor of 120 improvement - even if we could erk out a factor of 10 improvement, it would make things a lot better as far as the full userspace.

I'd really like to get a page in the "firmware/OS" section of the RBC wiki created for this - as well as the KISS-68030 page updated.

Subject: Re: Linux on KISS-68030 single board computer
Posted by dbdann on Sun, 28 Feb 2016 20:39:22 GMT
View Forum Message <> Reply to Message

Good work! I will *have* to build this board now.

Thank you!

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Sun, 28 Feb 2016 21:15:22 GMT
View Forum Message <> Reply to Message

>Will, is the Debian userspace you used new enough to have systemd?

Yes it is. Requires some additional kernel features -- I've built a kernel with those features, installed systemd, and it will boot the system. Did not time it (was busy with something else while it booted) but it still took a while. I had to modify the config a little as the default timeouts to wait for devices (90 sec) was too short. It also triggered some sort of non-fatal kernel bug late in the boot process; I need to look into that to understand it still.

There are some other options -- upstart, for example.

>I bet there are a lot of opportunities to get that full userspace boot time down

Yes, absolutely! A simple shell script could bring it up in a few seconds I expect.

Subject: Re: Linux on KISS-68030 single board computer
Posted by Andrew B on Sun, 28 Feb 2016 22:29:40 GMT
View Forum Message <> Reply to Message

Yeah, I'm not a huge fan of systemd in general.

I have so many boards in my pile to work on.... I'm not sure if I'll get to build one of these soon - but if there is anything I can do to help, let me know.

Subject: Re: Linux on KISS-68030 single board computer
Posted by Andrew B on Mon, 29 Feb 2016 03:52:16 GMT
View Forum Message <> Reply to Message

I did some reading, and it looks like busybox has a version of init and also "mdev" which can act as a replacement for udev.

https://github.com/hut/minirc
https://github.com/slashbeast/mdev-like-a-boss

It seems like to keep things as lightweight as possible, a busybox-based userspace might be a good choice?

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Fri, 04 Mar 2016 22:43:36 GMT
View Forum Message <> Reply to Message

Hi all

Quick update: I have the "secondary" MF/PIC board assembled. It provides UART and IDE interfaces only, interrupts are routed (through spare inverters) to the bus IRQ lines (I am using IRQ0 for serial, IRQ1 for IDE) and handled by the NS32202 on the primary MF/PIC. About a dozen chips on the secondary board can be omitted.

Both IDE and UART interfaces are confirmed working.

I have swapped the AHCT login on the bus transceivers for LS series. No noise issues observed (so far!) in combinations of bus slots which were previously problematic.

I had some issues with the interrupt controller; the timer would stop ticking when Linux booted. Resolved by increasing from 2 to 3 wait-states on I/O. I have not had this previously, unclear if the change to slower LS logic on the transceivers is related. I have enough chips to swap all back to AHCT so I may try that when I have the software issues sorted.

I have PPP up and running on the second UART; I can ping stuff on the internet etc. However some issues remain to investigate:

```
root@darkstar:~# wget sowerbutts.com
--2016-03-04 22:11:33--  http://sowerbutts.com/
Resolving sowerbutts.com (sowerbutts.com)... 195.149.84.65
Connecting to sowerbutts.com (sowerbutts.com)|195.149.84.65|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2188 (2.1K) [text/html]
Saving to: 'index.html'

index.html          0%[                    ]       0  --.-KB/s    in 0s
```

2016-03-04 22:11:33 (0.00 B/s) - Read error at byte 0/2188 (Invalid argument). Retrying.

Possibly wget is using some kernel feature I've compiled out, or maybe it doesn't like the fact that our timers have only 10ms resolution. Need to investigate. Telnet to port 80 on the web server and typing the request in by hand works fine so the TCP stack is certainly working.

I have issues with interrupt priorities; with the 1Mbit/sec serial port on IRQ0 (highest priority) it is possible to send data at 1Mbit/sec and overwhelm the machine. Similarly with second IDE on IRQ1 and timer on IRQ8 when you use the second IDE interface heavily about 60% of timer interrupts are lost.

I have looked in the NS32202 datasheet and I see it has several different priority modes including a "rotating priority" mode. Also there is the option of edge/level triggering. There is also interrupt nesting to consider. I need to sit down and figure out which modes would be best and how they would interact with Linux's IRQ handling.

It may be that the best solution is actually to have the timer as the highest priority interrupt, followed by the serial ports, with IDE last. This may mean some more hardware mods to re-arrange to fit the required order.

Can anyone confirm my ideas about the '202;
 - An EOI cycle (ie read with A13 set) simply clears the highest priority bit that is set in the ISRV register
 - The CPU can also clear the bit in ISRV itself by reading, masking, and writing back the register; doing this removes the need for a separate EOI
 - Writing (eg) 8 to FPRT will make IRQ8 the highest priority, so the priority order becomes: 8, 9, 10, 11, 12, 13, 14, 15, 0, 1, 2, 3, 4, 5, 6, 7 ie it just rotates the priority around -- the datasheet is unclear on this point.

Also can anyone confirm my understanding that we could run all the devices on this system in EITHER edge or level triggered modes without issue.

If anyone has thoughts on edge vs level, priorities, and the various modes of the 202, input would be most welcome at this stage.

Hmmmmmm. Lots to think about! But enough for tonight. Tomorrow I am in London all day so it will probably be Sunday before I get to look at this again.

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Mon, 07 Mar 2016 13:57:54 GMT
View Forum Message <> Reply to Message

Hi all

I have implemented both NS32202 hardware-controlled priority interrupts (any interrupt in service masks all lower priority interrupts), and a mode in which the kernel treats the NS32202 as a "dumb" level sensitive interrupt controller. One can choose between them at build time. I am using the hardware prioritised mode on my system.

I have re-arranged interrupts on my system very slightly - moving only IDE1. The NS32202 allows one to specify the highest priority interrupt, so I have made it IRQ10 which I have assigned to the timer tick. I have chosen to place the timer tick as highest priority, then any UARTs, then any disk interfaces.

With this change I no longer see lost timer ticks when the IDE1 port is in use.

The new IRQ allocation in my system is as follows:

```
 IRQ   DEVICE
  0    UART1
  1    4UART A (future)
  2    4UART B (future)
  3    4UART C (future)
  4    4UART D (future)
  5
  6
  7    IDE1        <-- moved (was previously IRQ 1)
  8
  9    IDE0
 10    TIMERH      <-- highest priority (then 11, 12, ... 15, 0, 1, ... 9 is lowest)
 11
 12    UART0
 13
 14
 15    TIMERL (unused. IRQ11 may be a better choice; software configurable)
```

I have implemented a mechanism for the kernel to read the NS32202 timer to get higher resolution time information. It now measures real time using the full resolution of the 1.8432MHz system timer.

I am back to 2 I/O wait states. I discovered that the 2A limit I had configured on my power supply was being exceeded now that I am using 74LS for the bus transceivers and have the secondary MF/PIC and additional CF card in the system. My power supply was therefore going into current limiting mode and the voltage to the system was dropping. Whoops! I think this is what was causing the NS32202 timer to drop out. I have raised the current limit to 3A which is sufficient and the system now appears stable at 2 wait states again.

I am still running the PPP link at 1Mbit/sec. I am pretty confident the system cannot sustain this data rate but have not had time to look into it fully yet.

Both UARTs are running with the receive FIFO threshold at 32 bytes. The threshold is runtime

configurable through a file in /sys. 32 bytes is probably the best choice for this system.

I have telnetd working on the system so one can telnet in over the PPP link. Works pretty well!

I have tried to get sshd working as well. It built the host keys (which takes nearly an hour!) and the daemon runs. One can get part way through the authentication process, but then the daemon dies with a SIGSEGV. Not quite sure why yet, but as the kernel also emits a stack backtrace at the same time I expect it may be some kernel feature I have configured out.

I am currently working on getting the KISS68030 support to "play nice" with the other m68k architectures so one can use a kernel that boots on all m68k machines on KISS68030. This will be an important step if we are ever to get support merged into mainline, which may be unlikely with fewer than ten machines in the world, but it will likely make it easier for us to maintain support out-of-tree also. Currently this effort has run aground with a null pointer dereference in the serial console driver. Ugh. I will keep hacking at it.

Will

Subject: Re: Linux on KISS-68030 single board computer
Posted by Andrew B on Mon, 07 Mar 2016 18:41:13 GMT
View Forum Message <> Reply to Message

I wonder if using the Dual IDE card that directly interfaces with the IDE devices without the 8255 (and also allows DMA) would speed up storage a bit?

Subject: Re: Linux on KISS-68030 single board computer
Posted by jcoffman on Mon, 07 Mar 2016 19:05:34 GMT
View Forum Message <> Reply to Message

Andrew,

It makes a difference on the Mini-M68k, but the speed difference between the 8mhz MC68008 and the 25-33mhz MC68030 with CACHE is considerable. I am impressed with the PPIDE speed on the 68030, and I like the speed of the Dual IDE on the Mini-M68k.

In all cases, the Dual IDE card would be faster.

--John

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Sat, 12 Mar 2016 15:17:37 GMT
View Forum Message <> Reply to Message

I have now documented the modifications required to construct a Secondary MF/PIC board which provides additional UART and IDE/parallel interfaces for this system.

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Sat, 19 Mar 2016 22:56:53 GMT
View Forum Message <> Reply to Message

I've made a second video. Shows off the faster boot process and networking.

https://www.youtube.com/watch?v=zsdBZ1Z29hs

Will

Subject: Re: Linux on KISS-68030 single board computer
Posted by adx on Sun, 20 Mar 2016 01:42:21 GMT
View Forum Message <> Reply to Message

Man now I'm really going to have to build one of these.

Subject: Re: Linux on KISS-68030 single board computer
Posted by Andrew B on Tue, 22 Mar 2016 06:25:35 GMT
View Forum Message <> Reply to Message

It's probably a pipe dream but it would be super cool to get the patches into mainline somehow and have an all-through-hole-component computer supported in the mainline kernel.

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Thu, 24 Mar 2016 17:54:33 GMT
View Forum Message <> Reply to Message

Andrew

Yes that would be great!

I fear it might be a hard sell to get the patches accepted though, given that there are fewer than 10 working machines in the world! It would be quite a lot of work for the m68k maintainers to take on maintaining another platform indefinitely given that they are already quite stretched.

I intend to get the patches to a standard where they would be good enough to accept even if the decision ultimately is to keep this platform out of tree.

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Mon, 28 Mar 2016 16:17:12 GMT
View Forum Message <> Reply to Message

Hi all

I have identified what appears to be a hardware flow control bug in the Linux 8250 UART driver, when handling the TL16C750.

I discovered the fault when I moved from a terminal emulator on a PC to my DEC VT510. The PC is very fast and uses a USB RS232 adapter with very deep (multiple kilobyte) FIFOs, consequently the KISS system's output is rarely (if ever) flow controlled. The VT510 is slow enough that, at 115200 baud, hardware flow control is required to avoid overruns.

The VT510 uses the DTR line to indicate when it has space in its input FIFO buffer. The VT510 DTR line is wired up to the CTS input on the MF/PIC UART.

Linux enables hardware-assisted flow control on the TL16C750, ie the chip is responsible for generating the outgoing RTS signal and monitoring the incoming CTS signal; the hardware will not begin transmitting a new character while the CTS line is asserted. When the CTS line is de-asserted it will continue transmission automatically.

In principle this should work perfectly, indeed most of the time it does. However when outputting a lot of text to the VT510 I find the output just stops after a while, apparently at random. Nothing further happens until I send some input to the terminal eg by typing a character -- then the output will continue, although it may halt again a short while later.

If I knobble the 8250 driver to disable this hardware-assisted feature, ie handle RTS and CTS just like a standard 16550, the issue goes away. Similarly if I use the VT510 with another serial device which can generate data faster than it can consume it (I tested with my PC) the issue does not occur. So it looks to be a fault with the flow control of output from the TL16C750.

I grabbed my scope and hooked up probes to:
- TX line (cyan trace; output from MF/PIC MAX232)
- CTS line (yellow trace; input to MF/PIC MAX232, note this signal is active low)
- UART IRQ line (magenta trace; input IRQ12 on NS32202 pin 5)

As usual deploying the scope quickly shed a lot of light on the problem.

This shows sending a bunch of data, then the eventual hang.

This is the same trace, zoomed in to show the last gasp. You can see the data transmission halts when the CTS line is asserted, but when it is later de-asserted the transmission does not resume.

This zooms in further on the last few bytes, and I think this reveals the issue: We take an interrupt from the UART to refill the TX buffer just as the CTS line is asserted. You can see here the scope decoding the last six bytes transmitted, "- 0038", which were indeed the last bytes to appear on the terminal. You can see the interrupt line asserted and released.

My guess at this stage is that Linux is reading out the CTS line in the interrupt handler, deciding to stop transmission because CTS is asserted, and expecting an interrupt when CTS is released. The hardware, meanwhile, believes that *it* is handling the CTS line, so it will not generate an interrupt when it is released.

Will do some digging in the driver source code and report back. I suspect the fix will be either: Disable auto-CTS when disabling the transmitter (but this may cause a race condition), or do not check CTS when auto-CTS is enabled.

Will report back.

Will

Subject: Re: Linux on KISS-68030 single board computer
Posted by will on Mon, 28 Mar 2016 16:51:40 GMT
View Forum Message <> Reply to Message

Well that was quicker to fix than I expected!

The fix is in and works perfectly. It is almost a one-liner. I will submit to the serial driver maintainer.

```
--- linux-4.5/drivers/tty/serial/8250/8250_port.c 2016-03-14 04:28:54.000000000 +0000
+++ linux-4.5-kiss/drivers/tty/serial/8250/8250_port.c 2016-03-28 17:55:45.580249260 +0100
@@ -2309,8 +2309,12 @@
   */
  if (up->capabilities & UART_CAP_AFE && port->fifosize >= 32) {
   up->mcr &= ~UART_MCR_AFE;
-  if (termios->c_cflag & CRTSCTS)
+  port->status &= ~UPSTAT_AUTOCTS;
+
+  if (termios->c_cflag & CRTSCTS) {
    up->mcr |= UART_MCR_AFE;
+   port->status |= UPSTAT_AUTOCTS;
+  }
```

```
}

/*
```

Subject: Re: Linux on KISS-68030 single board computer
Posted by ab0tj on Fri, 01 Apr 2016 20:08:22 GMT
View Forum Message <> Reply to Message

Thanks for all your hard work on this, Will. It is my intention to get Linux up-and-running on my KISS-68030, and then use it to host a BBS for fellow RetroBrew/RetroComputing enthusiasts. Sure, I could just run it on the Linux server I already have, but where's the fun in that?

Subject: Re: Linux on KISS-68030 single board computer
Posted by jvmartins on Thu, 09 Jul 2020 16:37:28 GMT
View Forum Message <> Reply to Message

Hi Will,

A few weeks back i was zapping on Youtube when i saw your retrobrew computer project which have caught my attention.
First of all, your project is Fantastic.
Took me a few days with your project going around my head. I had to try to do one myself.
Acquired the PCB's from Richard Cini, which already shipped them and hopeful i'll receive then in the next few days.

I know i have a nice journey in from of me, but i hope i can count on your support if i find some dead ends in the journey.

By listening your video i see that you used one MC68030RC33C at 33Mhz instead of the recommended 16 and 25Mhz in the documentation.
For that we need to alter the CPU oscillator to 32Mhz.

Did you altered any other component specification?

I started looking for the parts list for the 3 PCB boards (KISS-68030, MF/PIC board and the ECB back-plane).
Do you have record of your parts list that you could share with parts-numbers from any known vendor?

Regarding the old Motorola CPU i'm looking in Ebay for one but i'll wait for your reply before buying.

Kind regards,
Joao Vaz Martins
