

Re: Looking for suggestions for processor/module/sbc with ethernet

Source: <http://coding.derkeiler.com/Archive/General/comp.arch.embedded/2004-08/1768.html>

From: David Brown (david_at_no.westcontrol.spam.com)

Date: 08/25/04

Date: Wed, 25 Aug 2004 10:15:12 +0200

"Stephen" <spambox42@yahoo.co.uk> wrote in message
news:LKVniIATZ4KBFwPw@rtsoft.demon.co.uk...

>

> *Hello*

>

> *I'm looking for suggestions for an embedded microcontroller (processor,
> module or SBC all under consideration). I don't necessarily expect to get
> everything on one chip or module, we will be supplying our own motherboard
> either way, but the more we can get from the cpu/module then the less we
> need on the motherboard. The ideal list of requirements are roughly:*

>

> *- A fast 8-bitter, equivalent in performance to a theoretical 100MHz 8051
> with hardware multiply/divide unit, or a medium sized 16-bitter or a low
> end 32-bitter.*

>

I think you'd make your development life a lot easier by going for a cheap 32-bit chip – it will give you proper linear addressing over the whole range, plenty of speed for fast ethernet, and access to more ready-made network stacks and os'es. There are some very integrated ARM chips from Atmel (I've not used them, but our distributor keeps recommending them...), or something like the ColdFire MCF5234 (it has 100MB ethernet, a TPU unit which can easily give you 6 uarts and your pwms).

> *- 5V technology, or 3V with 5V compliant inputs/outputs. Surface mount.*

>

Forget 5V for the chip itself – you won't find many modern mcus with that sort of power that run directly from 5V. 5V tolerance is easy enough to make when needed, and 3V external memories are much cheaper and more available.

> *- 512KB code space. At least eprom, preferably ISP flash (serial onchip
> bootloader). Code estimate based on our current 8051 code which is 60000+
> lines in around 150KB using Keil C51.*

>

Most chips in your range should have jtag or on-chip debug, which can be used for programming.

Unless space is at a premium, there are a lot of advantages in external flash when you are looking at this size. Compare the prices of 32-bit micros with and without flash to see the cost of internal flash – you might pay five times as much for the internal flash. The situation is a bit different for manufacturers who are expert flash manufacturers who added a cpu, rather than expert cpu manufacturers who added a flash.

If you can get cheap ram (for example, the ColdFire supports sdram), then you can use the cheapest, slowest flash you can find – copy the program to sdram on start-up and run it from there.

> – *1MB battery backed up SRAM, although might consider 512KB SRAM if we can cost effectively also have an additional 512KB+ of flash, possibly compact flash.*

>

> – *Preferably a flat address space, but paging would be acceptable if we can get a "nice" page layout such as 32KB fixed + 32KB paged, and providing the development environment's requirements didn't steal lots of common page memory from us.*

>

> – *Ethernet. 10BaseT minimal, 100BaseT would be nice. Would consider a software solution or a single-chip solution, but either way, don't have time to write the TCP/IP stack.*

>

> – *50 to 60 general I/O pins.*

That's a lot of extra pins. If you can afford the board space, there is a lot to be said for something as simple as 74573 latches, or serial-to-parallel 74 logic chips.

>

> – *4 external interrupts. Multi-priority interrupt system.*

>

> – *2 CPU driven PWM channels.*

>

> – *6 serial ports, one of which can run in synchronous mode. (Suspect we'll end up with 2 plus a quad UART or 4 plus a dual).*

>

Or a ColdFire with TPU, which works fine as UARTs.

> – *SPI channel, although I'm happy to bit-bang that. Likewise I2C.*

>

> – *8x 10-bit ADC (multiplexing acceptable), although I suspect we'll probably need to add a chip for this.*

>

External is normally best for accuracy – you don't want to mix your high-speed digital signals and the analogue signals.

- > – *Enough CPU internal programmable timers that we have at least 2 free for our own private use after any other CPU components (e.g. serial + pwm) are driven from timers.*
- >
- > – *Realtime Clock, preferably a real hardware clock with programmable minute interrupts and alarm times, and not a software based interrupt driven clock off a simple "realtime counter".*
- >
- > – *Watchdog timer.*
- >
- > – *Power supervisor for RTC/SRAM protection and early power fail detection.*
- >
- > – *Low cost (of course!) at 100 off quantities (500/year max). Difficult to quantify given that our target is an overall price including our custom motherboard, the complexity of which depends on what the cpu/module/sbc gives us, but as a guide, a module with the processor/flash/sram should be under \$50.*
- >
- >
- > *We are currently considering either a fast top of the range 8051, the C166/167, or the Rabbit 3000 modules, but all of which have "issues".*
- >
- > *A top of the range 8051 seems to be prone to poor longevity. Not currently sure about the longevity of a C166 or the availability of non-royalty TCP/IP or how widely used the C166 is.*
- >
- > *The rabbit, although nicely priced with ethernet, currently looks like it hasn't got the horsepower. Dynamic C seems to suggest (in several documents) around 50000 lines of C takes 1MB, which compared to our 8051 running 60000 lines in around 150KB makes it around 6 to 8 times less efficient. Even accounting for the move to softools compiler (which we would definitely be doing), we are assuming some of the inefficiency comes about due to the Z80 architecture. Not only does this force up the code size, but we wonder how much gain the 44MHz really is when the code is this inefficient.*
- >
- > *Software-only based TCP/IP on a 8 bit is also a concern, not having done this before.*
- >
- > *So, if anyone has any alternative suggestions we can look into, I'd greatly appreciate it. I accept we are probably trying to get more out of a small 8 bit micro than most others would attempt, hence we're now considering 16/32 bit too.*
- >
- > *Stephen*