Re: How can I link the kernel with libgcc ?

Source: http://linux.derkeiler.com/Mailing−Lists/Kernel/2006−03/msg03476.html

- *From*: Ben Slusky <sluskyb@xxxxxxxxxxxxx>
- *Date*: Fri, 10 Mar 2006 13:41:05 −0500

On Fri, 10 Mar 2006 13:33:02 −0500, linux−os (Dick Johnson) wrote:

> On Fri, 10 Mar 2006, Carlos Munoz wrote:
>
> > Denis Vlasenko wrote:
> >
> > > On Friday 10 March 2006 05:47, Carlos Munoz wrote:
> > >
> > > > Lee Revell wrote:
> > > >
> > > > > On Thu, 2006−03−09 at 19:25 −0800, Carlos Munoz wrote:
> > > > >
> > > > > > I figured out how to get the driver to use floating point operations. I included source code (from an open source math library) for the

> log10
> function in
> the driver.
> Then I
> added the
> following
> lines to the
> file
> arch/sh/kernel/sh_ksyms.c:
>
> Where is the source code to
> your driver?
>
> Lee

Hi Lee,

Be warned. This driver is in the early stages
of development. There is
still a lot of work that needs to be done
(interrupt, dma, etc, etc).

What? You are using log10 only twice!

```
if (!(siu_obj_status & ST_OPEN)) {
...
/* = log2(over) */
ydef[22] = (u_int32_t)(log10((double)(over & 0x0000003f))
/
log10(2));
...
}
else {
...
if (coef) {
ydef[16] = 0x03045000 | (over << 26) | (tap − 4);
ydef[17] = (tap * 2 + 1);
/* = log2(over) */
ydef[22] = (u_int32_t)
```

(log10((double)(over & 0x0000003f)) / log10(2));
}

Don't you think that log10((double)(over & 0x0000003f)) / log10(2)
can have only 64 different values depending on the result of (over & 0x3f)?

Obtain them from precomputed uint32_t log10table[64].
−−
vda

Hi Denis,

Yes, the driver code so far only uses log10 twice, but there will be more uses for it as I populate the rest of the tables. However, I think its use will be some what limited. I wasn't aware that the floating point registers are not saved. I'll investigate a way to create a table with pre−calculated log10 values.

Thanks,

Carlos

Since the log in base n is the log in any base times a constant,
you can probably use log base 2 (binary bit position) and multiply
the result by a constant, which may simply be shifts and adds.

I assume you are using 16−bit audio. If so, the dynamic range
is only 20 * log10(2^16) = 96.3 dB. That means that attenuation
from mininum to maximum, in 1 dB steps, requires only 94 values.

Your code shows something whacked off at 0x3f = 0−>0x40 = 64
20 * log10(64) = 36 dB for only 36 values. Clearly, you don't
need floating point, just some thought ahead of time.

Cheers,
Dick Johnson

As Bart pointed out earlier, what this code is really trying to do is not log10(foo) but int(log2(foo)) for some positive integer foo. This can be simply expressed as fls(foo)−1 for all foo < 0. No floating point necessary.

−−
Ben Slusky | Trust is your enemy.

sluskyb@xxxxxxxxxxxxx | −Dan Farmer and
sluskyb@xxxxxxxxxx | Wietse Venema
PGP keyID ADA44B3B
−
To unsubscribe from this list: send the line "unsubscribe linux−kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxx
More majordomo info at http://vger.kernel.org/majordomo−info.html
Please read the FAQ at http://www.tux.org/lkml/