E1/T1 Hardware Interface - Feature #3237

Come up with GPS-DO design for E1 Interface

05/04/2018 10:34 PM - laforge

Status:	New	**Start date:**	05/04/2018
Priority:	Normal	**Due date:**	
Assignee:	vogelchr	**% Done:**	0%
Category:			
Target version:			

Description

We want to include a GPS-DO in our design, to ensure operation of the BTS at proper carrier frequency. That GPS-DO should have

- some GPS receiver with 1PPS output
- some local, voltage-controlled oscillator (VCTCXO)
  - the frequency should permit simple division to generate the 2.048 MHz required for E1 (otherwise we'd need a PLL)
- some DAC to steer the voltage-controlled oscillator
- some circuitry to count the number of of oscillator cycles between 1PPS edges
- some software to compensate for 1PPS jitter, loop filtering, possibly temperature compensation, etc.

See E1_Clock_Notes for more information.

Related issues:

Related to E1/T1 Hardware Interface - Feature #2484: Architecture of USB E1/T...	**Stalled**	**09/02/2017**
Related to E1/T1 Hardware Interface - Feature #3272: next-generation osmo-e1-...	**New**	**05/15/2018**
Related to E1/T1 Hardware Interface - Feature #3967: design a custom board fo...	**New**	**04/30/2019**

History

#1 - 05/05/2018 04:41 AM - tnt

The clocking architecture might be highly dependent on the choice of 'MCU' (sam3/xmos/ti/fpga...).

Most of these would have timer peripheral that will allow to do the counting, possibly even do the division to generate the 2.048M clock (if the vctxo clocks that 'mcu' directly) or have DAC, ...

#2 - 05/06/2018 09:00 AM - laforge

On Sat, May 05, 2018 at 04:41:56AM +0000, tnt [REDMINE] wrote:

> The clocking architecture might be highly dependent on the choice of 'MCU' (sam3/xmos/ti/fpga...).
>
> Most of these would have timer peripheral that will allow to do the counting, possibly even do the division to generate the 2.048M clock (if the vctxo clocks that 'mcu' directly) or have DAC, ...

I'm not sure any of those (except FPGA) are really suitable for the job? I guess you'd want a synchronous counter that really counts at the clock rate of the OCXO, without any clock conversion to some other clock internally which introduces significant jitter?

AFAIK, on both sam3 and possibly also xmos, all inputs are (re)sampled with some internal clock and hence I would expect related jitter.

Using something like the non-counter approach of the "Lars" GPS-DO, or something like http://www.leapsecond.com/pic/picdiv.htm to go from 10MHz t 1 pps (and then simply compare the phase of the two 1PPS with each other?) would not suffer from that.

But then, maybe the jitter is tolerable, I don't know...

In the ideal world, we would come up with some GPS-DO that's capable of generating the 2.048 MHz irrespective of what we use as processor for the E1. This way we stay open to multiple approaches.

#3 - 05/06/2018 10:16 AM - tnt

I was thinking of two options (that I'll illustrate using the SAM4S here, didn't check XMOS):

1. Timers have an external clock input pin and feed that with the VCXO. I actually thought that this would synchronously feed the counter directly,

but you're right, it's actually re-synced to the internal peripheral clock. Now, I don't think it would be *that* much of an issue because:
   - The peripheral clock is going to be much higher in any case * By progressively increasing the time constant of the software filtering (to get quick lock at the beginning, but be more stable after), that jitter wouldn't matter much I think * In any case I don't think the jitter of the clock we generate any way is all that important (well, needs to be reasonable of course and withing E1 spec), because the BTS is going to have its own oscillator and will just lock the long term frequency to what we send.

1. But any way, a better option I would think is to clock the MCU itself with the VCXO as well. So for instance for the SAM4S:
   - You'd use a 30.72M VCXO and feed that to the SAM4S * Using the internal PLL, you can divide it by 8 and then multiply by 25 to generate the 96 MHz output needed for USB. * Using the other PLL you can generate the core clock, something like divide by 3, multiply by 10 to generate a 102.4 MHz clock. * Using a PWM peripheral it should be possible to output 2.048 MHz that's perfectly synced to the input 30.72M * Use the TC to count pulse between PPS and drive a DAC, or even possibly use a heavy filtered PWM channel.

#4 - 05/06/2018 10:53 AM - vogelchr

If we do it all within a microcontroller, and to receive E1 bits we probably want to sample the incoming signal at a rate ≥4×f_E1 (pulses are ~250ns), so we would need two synchronized SPI blocks at 8,192 Mhz. Same holds true for generating the pulses.

Otherwise the idea of running the MCU from the VCXO seems to be a good idea.

#5 - 05/06/2018 07:13 PM - laforge

- Related to Feature #2484: Architecture of USB E1/T1/J1 adapter added

#6 - 05/17/2018 08:48 AM - laforge

- Related to Feature #3272: next-generation osmo-e1-xcvr board added

#7 - 05/17/2018 08:48 AM - laforge

- Assignee set to vogelchr

#8 - 05/22/2018 05:22 PM - vogelchr

- File sam4s_clock_pll_scheme.pdf added

- File sam4s_clock_pll_scheme.svg added

I've drawn up the clocking scheme proposed by tnt on top of a page of the SAM4S user manual. The programmable clock controller can output the 2,048 MHz for the LIU directly to a pin. That way we use up both PLLs and we will run the core at 96 MHz.

#9 - 05/30/2018 03:02 PM - laforge

- Tags set to E1

#10 - 06/01/2018 09:09 PM - vogelchr

SAM-BA over USB works for an 18.432 MHz XTAL (9*2.048), but the boot code seems to expect a crystal, not an external oscillator which has different input requirements on the signal fed to XIN. (§24.2, Note 1). We will have to attenuate the 3.3V crystal oscillator output to not overdrive the xtal input amplifier.

If 18.432 MHz oscillators are unavailable, we could add a separate one only for SAM-BA, and alternate between them outputs via the oscillator enable pins.

#11 - 06/01/2018 09:50 PM - laforge

As indicated, I think having SAM-BA only on UART is not such a big issue. We add
the usual 2.5mm jack, and people can use their "Motorola T191" aka "OsmocomBB" serial
cables.

#12 - 04/30/2019 09:33 AM - laforge

- Related to Feature #3967: design a custom board for the ICE40 E1 adapter added

Files

sam4s_clock_pll_scheme.pdf	227 KB	05/22/2018	vogelchr
sam4s_clock_pll_scheme.svg	111 KB	05/22/2018	vogelchr