Si5351A: facts and myths

Hans Summers

6/28/17 #28999

I'm forwarding this to the BITX group as someone suggested it may be interesting to some here!
73 Hans G0UPL
http://qrp-labs.com

---------- Forwarded message ----------
From: **Hans Summers** <hans.summers@...>
Date: Wed, Jun 28, 2017 at 10:15 AM
Subject: Si5351A: facts and myths
To: emrfd@...

Hi all

Forgive the long post please. I have extensive experience of the Si5351A now as many of the QRP Labs products are based around this miraculous little chip. So to some extent, I do know something about what I am talking about here ;-)

DDS vs PLL

This confuses a lot of people so just to clear this up at the start. DDS is Direct Digital Synthesis which simulates a sinewave by accumulating numbers in an internal register at regular intervals (determined by the system reference oscillator e.g. often 125MHz for the simple AD9850 DDS modules that are common). This accumulated count goes to a Digital To Analogue converter (DAC) in the chip, and through an anti-aliasing filter (which you must add external to the chip). Due to the discrete stepped nature of the simulated output sinewave, the frequency spectrum has spurs and this is a feature of DDS. Higher ratio of reference to output frequency, and wider DAC bit width both reduce the spurs. Analog Devices has market dominance of DDS chips and some very expensive high end chips as

well as the more common cheaper ones.

The SiLabs Si5351A and Si570 chips implement a totally different Phase Locked Loop (PLL) based method of synthesising arbitrary frequencies. The Digital PLL is all inside the tiny 3 x 3mm chip and replaces an entire circuit board of earlier synthesisers some decades back! The chip multiplies a 27MHz (or 25MHz) reference up to an internal Voltage Controlled Oscillator (VCO) in the range 600-900MHz, which you can then divide down to get the desired output frequency. Any PLL has the characteristic of adding phase noise to the crystal reference frequency. How much phase noise is added depends on many factors, particularly the loop bandwidth - in the Si5351A and Si570 the loop bandwidth is low (slow loop) and this really makes the phase noise very low.

I just mention this because a lot of people seem to mistakenly refer to the Si5351A as a "DDS" which it totally is not! And also to be clear that a feature of DDSs is spurious outputs (like birdies in a receiver) whereas a feature of PLLs is phase noise. Nothing is for free! In modern devices I think that for the majority of applications both of these undesirable features are usually negligible.

Upper frequency limit:

The upper frequency limit of the Si5351A-B chip is 200MHz. Not the widely circulated 160MHz. The 160MHz applied to the earlier chip revision. The QRP Labs Synth kit http://qrp-labs.com/synth uses the current -B revision (200MHz limit) and I expect any other hobby boards using this chip are also using the -B revision of the chip.

Practical frequency limits:

The practical frequency limits are even wider than the datasheet suggests. The datasheet says the internal VCO frequency should be 600-900MHz. If you are prepared to disrepect this specification, then who knows what performance parameters may be affected, but it DOES at least seem to work... bear in mind that in any case, some performance parameters degrade with frequency in any case, so your own frequency limit might be lower. The configuration registers of the Si5351A can be configured to allow operation down to 3.5kHz. The upper limit, determined here by measurement, was approx 292MHz. Other QRP Labs kit builders have also confirmed similar figures. Beyond 292MHz my Si5351A simply stopped producing an output. The cut-off was quite sudden.

Phase noise compared to Si570:

Perhaps one of the biggest (probable) myths, is that the Si5351A has high phase noise so we shouldn't use it if we care about noise! Of course every application does have different

requirements and trade-offs are always present. Base the trade-offs on reality though not on rumour. The Si5351A and Si570 might be said to come from the same family of digital PLL chips but they do have a lot of differences.

The Si5351A multiplies the 27MHz (or 25MHz) crystal reference up to an internal PLL frequency of 600-900MHz. Then it is divided down in the so-called "Multi-synth" stages to produce the desired output frequency (or frequencies). Both the multiplication up, and the division down, are fractional divisors - an integer and a fractional part made of 20-bit numerator and denominator.

The Si570 has an integer-only division ratio down to the output frequency. The Si5351A has a fractional division ratio. The Si5351A datasheet recommends using even integer divisors to minimise output jitter (a.k.a. phase noise). This makes sense! Think about dividing by 2 a few times, it is a relatively clean process. On the other hand if you divide by 45 and 132,448 / 382,124ths for example, you can imagine that the fractional division will result in a messy output pulse stream. In other words, much worse phase noise.

The jitter specifications in the Si5351A datasheet are noted to be "worst case, real world" and it says they "Jitter is highly dependent on device frequency configuration". We don't know what configuration they used and whether they used even integer or not. The "worst case" note seems to imply not. SiLabs documentation is quite poor in some respects.

I'm saying you can't just compare datasheet jitter specifications... a lot of evaluation and measurement would probably be needed side by side to determine how the Si5351A really compares to the Si570 in actual use.

All the QRP Labs firmware (and example source code) use the minimum jitter even integer divider mode.

Note also that Elecraft use the Si5351A in the KX2, not the Si570. Everyone knows Elecraft make great radios. So there's a fair argument that what's good enough in an Elecraft rig cannot be too bad ;-)

Both an Si5351A and an Si570 would be worse phase noise than a crystal... but then again, a crystal isn't movable over 3.5kHz to 292MHz... An interesting article on the QRP Labs website by Gwyn G3ZIL about comparing the Si5351A to a crystal LO in a practical receiver. The article is interesting not just because of its conclusion that an Si5351A does a great job for practical real world purposes, but also because it stresses the importance of clean supplies when using the Si5351A. Noise on the supply line does easily modulate the Si5351A output. So keep that in mind when designing around the Si5351A. See http://www.qrp-labs.com/synth/synthnoise.html

The controversial PLL reset and audio "pops":

Yes, when you do a PLL reset, it interrupts the output frequencies for some milliseconds and that will make a nasty "pop" in your audio. This PLL reset is one of the most misunderstood aspects of using the Si5351A. For good reason! The SiLabs documentation is poor and does not explain clearly when you need it and when you do not! I know that there is the knowledgebase article http://community.silabs.com/mgrfq63796/attachments/mgrfq63796/Timing_Discussion%40tkb/61/1/311668.pdf dealing with the topic, which says a PLL reset is required every time the PLL feedback registers are changed. This document also refers to another KB article 311538 which when you open it, mysteriously contains exactly the same contents as 311668. Which is typical of the Si5351A documentation - even the AN619 (choosing a register map) contains lots of errors and duplication caused by copy-and-paste without subsequent necessary edits. I don't want to be too harsh because I know only too well it is quite a big job, keeping documentation all in order.

When you first set up the Si5351A registers the first time, you need to do a PLL reset otherwise nothing works. You do NOT need to do a PLL reset for subsequent PLL feedback divider changes! I have not been able to find any size of frequency change where a PLL reset is required. Certainly incrementally tuning across an amateur band requires no PLL reset. Given the datasheet comments and the fact that evidently you do at least need one PLL reset at the start, I remained a bit nervous and in my code I do generate a PLL reset when there is a "large" frequency step... what does "large" mean, who knows. I arbitrarily set it to 10kHz and no QRP Labs customer has ever had a problem with it to the best of my knowledge.

What the datasheet does NOT tell you, is that if you wish to use the phase offset feature to maintain an accurate and constant phase offset between two of the Si5351A outputs on the same frequency - then you MUST do a PLL reset under two circumstances:

1) Every time you change the MultiSynth divider (whether integer or fractional) - note, the MutliSynth divider! Not the PLL feedback divider! Which is kind of opposite to the documentation... but trust me, this is the way the chip actually works

2) Every time you switch the outputs on/off using the Clock Control registers e.g. Register 16. If you are about phase relationships, INCLUDING if you have simply set a 180-degree phase relationship to another clock output using the clock invert bit in the clock control register, even then, you have to do a PLL Reset.

If you don't do the PLL reset under these two circumstances then the phase offset will be

random. Frequency will be fine though - this part of the PLL reset discussion only applies to when you wish to have a precise phase offset between outputs. Examples which I use are when you want push-pull outputs (180-degree phase difference), some people use this for driving a LF PA; or when you want 90-degree phase offset for switching a Quadrature Sampling Detector type mixer which requires a quadrature LO.

I have not seen any of this anywhere in SiLabs documentation, I reached the above conclusions after hours and hours of experimenting...

It is very interesting to me that the few circumstances in which a PLL reset is actually required, apparently are related to the use of the MutliSynth Divider stage of the chip - not the PLL! It almost makes me wonder if this PLL reset register is totally misnamed and totally misunderstood - should it really be called the "MultiSynth Reset Register"?

Difficulty

There is a view that programming the Si5351A is difficult - or perhaps, much more difficult than a DDS chip. I would agree that it is more difficult conceptually than loading a 40-bit tuning word into a DDS. But not MUCH more difficult! Sure, you have to get your head around the Si5351A. You had to get your head around the DDS too, it's just that the DDS has been around for longer and we have got more used to it.

In either case, the internet is now full of examples and libraries of how to use both families of devices, so you can easily use either without much deep understanding of how they work. QRP Labs has example code for AVR, PIC and Arduino http://qrp-labs.com/synth and is just one of many sites publishing source code for the Si5351A.

The above discussions on PLL reset maybe sound scary but remember, the only reason we can discuss all this is because the Si5351A has a much greater capability and feature set in many ways, then a DDS. The multiple outputs for example, which can be on different frequencies or different phase shifts.

Requirement for TCXO

A TCXO has two benefits - first, as it is a Temperature Controlled Crystal Oscillator, it exhibits very low temperature dependence compared to an unstabilised crystal oscillator. The second benefit is that if you buy a 27MHz TCXO then without any further discussion, when you just power it up, the output frequency will be much closer to 27MHz than a crystal oscillator would be. For example we find that in the QRP Labs synth the 27MHz crystal typically operates at around 27.004MHz +/- 1kHz. So if you want an accurate synthesiser, then you need to measure this value and compensate for it in the Si5351A register

configuration (easily done in firmware).

For all but the most demanding applications, I think the extreme frequency stability is not really necessary; and in many cases measurement (a one-off calibration) is possible so a regular crystal does just fine. Even in very demanding applications like weak signal narrow band modes (e.g. used in the Ultimate3S transmitter http://qrp-labs.com/ultimate3s/u3s) a basic crystal has been shown to be good enough across HF and even VHF all the way up to 2m. QRP Labs App Note AN001 http://qrp-labs.com/appnotes discusses some techniques to deal with the drift - without a TCXO.

Bear in mind also that many of the less expensive TCXOs use a digital compensation method, where they measure the temperature digitally then apply a correction to the pull the crystal oscillator. In many cases the result is discrete steps on the output frequency, of several Hz. That can be enough to disrupt any weak signal narrowband work!

So just weigh all this up before automatically assuming you need a TCXO. The QRP Labs Si5351A Synth kit http://qrp-labs.com/synth has PCB footprints for a TCXO if you wish to use it, but personally I have not found it to be necessary.

Conclusion

So I hope this helps someone... when I started writing this I was only going to comment on the PLL Reset topic, and on the fact that the current chip revision has a 200MHz specified upper frequency limit. But I ended up adding a lot of other info.

73 Hans G0UPL
http://qrp-labs.com
http://hanssummers.com

4 People liked this

Jerry Gaffke 6/28/17 #29001

Good post.

A few comments:

The output dividers in fractional mode are indeed dividing by two integers as you say, which

normally creates lots of jitter. The PLL has its loop filter to suppress such jittter. But the trick about msynth dividers is that they use digital logic to predict how far off from ideal each edge is, and then use programmable delay lines to put the edge in exactly the right place. Jitter is higher, but (as I recall from looking at Si5338 docs a couple years ago) is not very much higher than integer mode. We're using the Etherkit library on the Raduino which uses fractional output divides except at frequencies over 112mhz. This does make programming easier (my si5351bx routines posted here earlier today take about 30 lines total of code), and makes it possible to have three fully independent clocks. There are three output msynth dividers, but only two PLL's.

The part has been out for years, phase noise must be low enough if nobody has yet figured out definitively if and when and how much the phase noise is an issue.

Regarding TCXO's, would be interesting to park a temperature monitor (a diode?) near the 25mhz crystal, do our own TCXO computations inside the Nano with very fine grained frequency changes. But I'm not sure that's reasonable to do on the Si5351, see next issue.

It takes 8 i2c register writes to change frequency (pll or output msynth), and at 10us per bit that's around a millisecond. Any idea if they buffer those writes, or do we see dirt as each of those register writes takes effect?

Pavel here reports that his library can run the Si5351 at 225mhz. Makes sense, that's 900mhz/4. I wonder if the VCO can be pushed beyond 900mhz. But outputs are CMOS only, that could be a good part of why their max freq spec is so conservative.

When I googled for "Si5351 PLL reset needed", I found several hits where somebody found updatating the PLL did require a PLL reset. Though would be curious if it did turn out to be not resetting the PLL at all. But I can easily imagine that a 600-900mhz vco would require a bit of mothering on the loop filter.

Jerry Gaffke 6/28/17 #29002

I guess you did too. Hmm. 292mhz.

Show quoted text

Jerry Gaffke 6/29/17 #29003

Check out the Si5341, same family as the Si5351 but higher performance.
And a more complete datasheet.

Jitter specs look better than an Si570, even when in fractional output msynth mode.
They spec fractional output divide mode to have about 20% worse jitter than integer output divide mode.

Jerry

Show quoted text

Ashhar Farhan 6/29/17 #29004

i have been so bogged down with the ubitx push that i havent gotten to evaluate the Si5351 with an external, low noise source. I would imagine that a better built oscillator in place of the internal oscillator with lower phase noise will result in a better phase noise from the Si5351. The notch filter is lying somewhere, i need to dig it out.

- f

Show quoted text

John Backo 6/29/17 #29005

Jerry:

The Si5341 has better specs. But it is a VQFN package with
64 smd pins to contend with. That is probably a bit much for this group.

And the unit price is about $US16.00 at Digi-key. Sounds good, but...
(and it may be discontinued.) And the evaluation board is $US200.

Hi.
john
AD5YE

Jerry Gaffke 6/29/17 #29006

Oh, I'm not suggesting we use the Si5341.
Just saying that it's a similar part, and that it shows fractional output msynths can give nearly as good a jitter spec as integer output msynths.
Only about 20% worse.

The Etherkit Si5351 library we currently use has fractional output msynths.

If we stick with linear regulators off of 12v, powering the Si5341 would suck more watts than we are putting out the antenna.

Jerry, KE7ER

Show quoted text

Hans Summers 6/29/17 #29008

Hi Jerry

Thanks for the info on digital logic fixing the edges with programmable delay lines, very interesting.

In my Ultimate3S kit http://qrp-labs.com/ultimate3/u3s and VFO kit http://qrp-labs.com/vfo and ProgRock kit http://qrp-labs.com/progrock I use the Si5351A in the even integer divider mode. I don't find it troublesome adjusting the PLL. I have evolved my calculations so that now I have everything done using 32-bit integer arithmetic, without loss of precision. Removal of floating point saved a lot of code space. By the way if you use 64-bit integer arithmetic the code space is even higher than if you use floating point.

> It takes 8 i2c register writes to change frequency (pll or output msynth),
>and at 10us per bit that's around a millisecond. Any idea if they buffer those
>writes, or do we see dirt as each of those register writes takes effect?

For 8 bytes, if they take 1ms each, that would be 8ms.

10us per bit assumes 100kHz I2C bus. The I2C bus can run at 400kHz and still be within spec. That reduces it to 2.5us per bit.

Not all 8 I2C registers need to be written to change the frequency. When you actually look at it in great detail you find that you can arrange the configuration so that only around HALF of these eight I2C registers actually

change. That doubles the write speed again.

The Si5351A does NOT have buffering, as far as I know. If it did, I think they ought to celebrate that on page 1 of the datasheet... it would be *nice* indeed if it did, but we have to remember this thing was intended as a programmable clock generator, not for amateur radio VFOs specifically ;-)

Intriguingly the Si5351A datasheet writes right there at the top of page 1, "Glitchless frequency changes". Then makes no further mention of what this means. <groan>

The datasheet also does not specify the settling time upon making a frequency change, the only clue is the power-up time specified as 2ms typical, 10ms maximum.

So I think the answer is that dirt could reach the output while each register write takes effect. This is something I still want to experiment with and find out. Is it better to alter the PLL frequency, or the MultiSynth divider, from this perspective? It would be interesting to experiment more with it. One might perhaps think that writing to the MultiSynth divider would take immediate effect, therefore as each register partially writes something into the divider logic, it would immediately produce dirt at the output temporarily until all the other registers are written. Whereas if you write to the PLL divider, if the PLL takes LONGER to settle on its new frequency due to the finite PLL loop bandwidth, then perhaps it is more immune to partially written configurations. I mean if it takes a couple of milliseconds to write all the registers, and this is faster than the loop can react, then the actual VCO frequency might be relatively more immune to dirt during configuration writes, than when writing the MultiSynth dividers. Just SPECULATING. Yet another interesting experiment would be to see if there is any benefit to writing the configuration registers in a particular order.

73 Hans G0UPL
http://qrp-labs.com

Jerry Gaffke

6/29/17 #29014

A possibility for mostly cleaning up transition dirt on the Si5351 would be to pingpong between two VCO's or two output msynth's
https://groups.io/g/BITX20/message/28900
On the other hand, a sudden switch between clocks might be worse in some cases than a smoothly skewing vco.

The code I posted yesterday has a testbench where you can listen to the Si5351 move through a range of frequencies. I was using a scratchy tuning pot, any cruft I heard seemed

to correspond with a jump in frequency displayed in the Arduino uart monitor, turning the pot slowly it sounded just like an analog VFO transitioning. But my notes in the code suggest there might be worse at msb transitions.

Not much testing on that code, nothing more than the testbench suggests. And I don't have an accurate frequency counter. I really have to pull my head out of this and pay attention to some other matters. But if anyone finds what they think is a bug, the best way to report it would be to send me the Si5351 register dump and the state of the global RAM variables and the args to si5351bx_setfreq()

The si5351bx code I posted yesterday uses the downshift trick described here:
https://groups.io/g/BITX20/message/28768
So to calculate the a,b,c in fvco/fout == (a+b/c) takes one 32 bit integer divide (I assume the compiler is smart enough
to do the divide once and harvest both quotient and remainder) and some downshifts.
The correction trick for c described there can give extremely tight results if you are willing to burn some 64 bit integer math.

Jerry

Show quoted text

Jerry Gaffke

6/29/17 #29018

One more tidbit.

In that old post on the downshift trick, I was moaning about choice of c:
> One curious thing I see: the "c" of (a+b/c) was getting set to 0xFFFFF by the Etherkit code,
> now it is set to an even million so 0xF4240. Pavel's code sets it to 1048574, which is 0xFFFFE. Why not 0xFFFFF?

As mentioned on p10 of AN619, using that fancier Si5351 in VCO mode requires a value for c in the feedback msynth of 1000000.
Mystery explained.

Jerry

Show quoted text

Dr Fred Hambrecht

6/29/17 #29021

Query Jerry,

I have not looked at the internals of the Si5351 but wondered if it is possible to replace the 25 MHZ crystal with a 10 MHZ signal and of course change the divide ratios? I am referring to the breakout board as opposed to just the chip. I know the SI5351C has the ability to use an external reference, but no one makes a breakout board with it. I am presently guiding the raduino with the 1PPS signal from a GPS module but want to be able to use my 10 MHZ GPS disciplined oscillator.

v/r
Fred W4JLE

Show quoted text

Jerry Gaffke

6/29/17 #29024

As you say, the Si5351C can take an external reference from 10 to 100 mhz or so through a specific pin.
I had once thought we could just jam 10mhz into one of the crystal pins on our $1 MSOP, but Hans corrected me
in another group (I think emrfd). The crystal pins on the MSOP are tuned for 25-27mhz, don't work well outside that range.
Either don't work at all, or too much jitter.

Am curious exactly how you are using that 1pps GPS to guide the raduino. So you now have an accurate Si5351 clock,
perhaps running a Nano counter-timer from a spare Si5351 clock at a few khz, and capturing counts with the pps?

Jerry

Show quoted text

Jerry Gaffke

6/29/17 #29027

From post 28768:

##################
Here's python code for an algorithm to compute (a+b/c)I haven't seen elsewhere. All values are integers, the divides need to handle 64 bits.of dividend. Without the final correction to c, I get slightly better accuracy than I do with the downshifting trick of the code previously posted. This is about the same algorithm as what's in the Etherkit library. With the final correction, it is five times better.

```
    c = 0x100000-1
    a = v/x
    b = ((v%x)*c)/x
    c = b*x/(v%x)
```

##########################

That correction scheme might not be obvious.
First off, note that if b=1 and c=1000000, we can double the size of b/c by either increasing b by 1 or by decreasing c by 500000.
Obviously we have a lot more granularity if we start messing with c.

In the case of the pll feedback msynth we want non negative integers a,b,c with c greater than b such that (a+b/c) == fvco/fxtal.
Same applies to an output msynth, except the ratio is fvco/fout.
We start with c =0xfffff, and take our best shot at b, but rounding down so that b/c is less than or equal to the desired fractional ratio.
The ratio we really want for b/c is the fractional part of fvco/fxtal, or (fvco%fxtal)/fxtal
where % is the c operator to find the remainder of a divide,
and the "/" we can assume for now is done in floating point.
So we want (fvco%fxtal)/fxtal == b/c   where both "/" operators are assumed to be done in floating point.
Solving for c we first invert the equation and then multiply both sides with b, getting
 fxtal/(fvco%fxtal)*b = c
Since fxtal is much larger than fvco%fxtal, we can now forget about floating point and do that as an integer divide.

The "five times better" is what I was seeing in my python simulation of the calculations, I'm not quite sure why it was only 5x.
It could be much more accurate than that.
But we're now talking just a few ppb.

Jerry

Show quoted text

Jerry Gaffke 6/29/17 #29029

OK, I've now thought a little bit harder.

We can get much much more accurate in the case of initial values of 1/0xFFFFF for b/c.
But not if the initial values are 0xFFFFE/0xFFFFF.

So may help in some cases, but not all.
In the case of a GPS disciplined Si5351 we could choose a vco frequency with a low initial b value.

Jerry

Show quoted text

Dr Fred Hambrecht 6/29/17 #29044

I am using the method illustrated by Gene Marcus in his QEX article. I count a 2.5 MHz signal from CLK0 for 40 seconds determined by a 1PPS signal from the GPS module ($6.00 on Ebay) then divide by 4 and subtract from 25MHz and update the correction. His article is available from http://www.arrl.org/files/file/QEX_Next_Issue/2015/Jul-Aug_2015/Marcus.pdf

I am using my code but his ideas. I am also using a 20x4 display. In addition to being able to operate all modes and bands (I am using a filter board from a Codan NGT, Ebay $5.07)
I display VFO freq, mode, Maidenhead grid location (thanks to Jack for the code improvement), date and time, and altitude. Much of Gene's code for handling the GPS module was replaced by the TinyGPSPlus library.

It looks like I will have to use the 10MHz to control a 25 MHz phased lock loop.

I am loving the education the $59 tuition has provided! I am taken back to my beginning ham days in 1956 and ham radio is once again fun!

Thank you Sir for the information, it is much appreciated.

v/r
Fred W4JLE
Show quoted text

Jerry Gaffke 6/29/17 #29071

Hans,

I took the code from https://groups.io/g/BITX20/files/KE7ER/si5351bx_0_0.ino
and replaced the final while loop with this:

```
    // Slews CLK2 from 6999950 to 7000050 hz and back
    // advancing in 1 hz increments once every 5 milliseconds.
    // So takes 1 second for each round trip.
    // When going up through 7.0mhz we have a transition in
    // the output msynth from a=126 to a=125, since 875/7.0 = 125.0
    // Would expect to hear a tick there since those 8 msynth registers
    // are not all updated simultaneously
    while (1) {
        for (n=0; n<100; n++) {
            si5351bx_setfreq(2, 6999950+n);
            delay(5);           // 5ms
        }
        for (n=0; n<100; n++) {
            si5351bx_setfreq(2, 7000050-n);
            delay(5);           // 5ms
        }
    }
```

Note that the delay() function call is susceptible to exactly when 1 ms interrupts occur
and to just how long it takes to execute my code. So likely sometimes an extra ms in there.

Listening in with a nearby Bitx40 I hear no hint of a tick. Just a cleanly varying tone.
Time to celebrate.

Jerry, KE7ER

Show quoted text

Axel DK4AQ

6/29/17 #29072

Hi all,

I developed an own VFO basing on Si5351 and Arduino in 2015. I had the same question: Is the Quality of the oscillator signal good enough for a high quality SW receiver ?

Searching for information I found a blog of NT7S „Ripples in the Ether“ http://nt7s.com/ . Search under „Si5351“ and look at „Si5351A Investigations Part 7“. He organized measurement of the phase noise of the Si5351 in different configurations. The measurement was done with professional equipment by KE5FX. He compared the results with data from Sherwood Engineering receiver table, containing phase noise of LO of well known HAM equipment.

After omparing the values, I felt I should not have any headache about the signal quality of the Si5351 ! Thanks to NT7S and KE5FX for their helpful informations !

Only one little problem with Si5351 stayed. Using switched mixers it makes sense to use a rectangular oscillator signal. Looking to the spectrum of rectangular signals, you will find all harmonic frequencies (fo, 2fo,3fo...). That means if you have a receiver mixer you will receive signals from all these frequencies +/- IF. In case of transmitter mixer you will send on all these frequencies at the same time ! Ususal RF filters have to reject them. If we chose an exactly symetric rectangle (50% duty cycle) then all even harmonics have zero amplitude. So at least the nearest harmonic frequency 2fo is rejected and the filters have to be less complex. Speaking about birdies it is also useful to avoid some harmonics.
Unlucky most digital controlled oscillators cannot deliver exact 50% duty cycle at every output frequeny. A good idea is to use a toggle flipflop at the output signal, dividing the output frequeny by 2. The output signal is a good aproach to 50%, there stay some very little unsymetric parts resulting from noise on the switching levels inside the flipflop. To double the output frequency at Si5351 is a simple task.

By the way : I had no noises with tuning. The frequency changed smoothly without problems.

So enjoy working with Si5351 and BITX40 !

Axel / DK4AQ

Jerry Gaffke 6/29/17 #29078

That might be a good argument to use the R output divider after the output multisynth. Would be interesting to see if a divide by two in R generates less of a second harmonic on the Bitx40, just a matter of changing the raduino sketch. My guess is that it won't matter much. Given how low the Si5351 jitter is in fractional mode, they probably do a pretty good job on duty cycle as well. I'd bet non-linearities around the IRF510 would swamp any such effect. Though when it comes to the uBitx push-pull PA I'm not planning to bet a lot.

Jerry, KE7ER

Show quoted text

Hans Summers 6/30/17 #29088

Very interesting, Jerry

Although there is no hint of double-buffering in the datasheet - maybe there really IS something there inside that chip?

It would be very interesting to slow everything down so that the I2C update of the 8 registers takes 5ms. In other words - remove the delay(5), and perhaps have an output pin toggling so that you can hook on an oscillscope and see how often the output frequency is being updated. Then put delays in the I2C write - so that each step takes 5ms. It would be very interesting to see what happens then!

73 Hans G0UPL
http://qrp-labs.com

Jerry Gaffke 6/30/17 #29097

Yup, delays on those i2c writes could be revealing. Occurred to me but I haven't gotten around to it. But I did put a scope on it, looked at the 7mhz waveform with 5ms of delayed

sweep after triggering on an edge. Seemed clean. The important thing for me is it sounds clean on the SW receiver, even when going through a transition between one integer and the next.

On p20 of the Si5338 datasheet https://www.silabs.com/documents/public/data-sheets/Si5338.pdf
they quickly mention how they clean up those edges. I'd guess the mechanism is the same as the phase adjustment they make available to us, which is not infinitely accurate.

> To eliminate phase error generated by this process, the MultiSynth block calculates
> the relative phase difference between the clock produced by the fractional-N divider
> and the desired output clock and dynamically adjusts the phase to match the ideal clock waveform.

In the next paragraph they mention how it's good for any frequency up to vco/8.0, and then integer divides of 6 and 4. Just like the Si5351. At $10, the Si5338 was apparently able to afford a tech writer. So the Si5338 docs might be a good way to fill in holes left by the Si5351 docs.

Jerry, KE7ER

Show quoted text

← Previous Topic Next Topic →

1 - 20 of 34 < 1 2 >