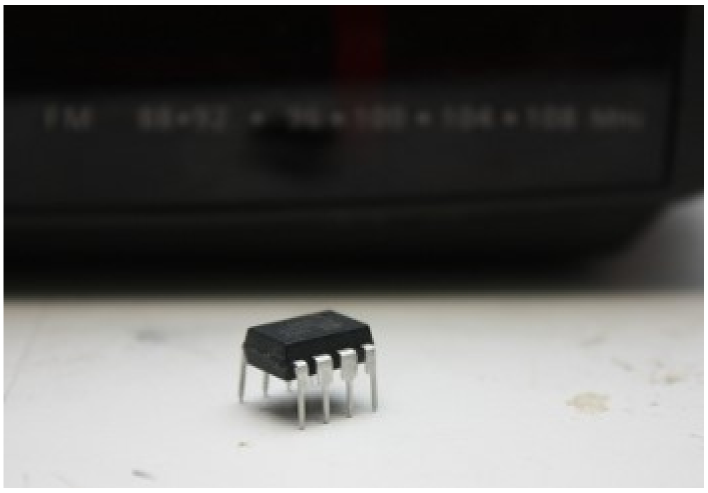
SpritesMods.com

AVR-based FM-transmitter - Intro, theory

Intro, theory



I think I may have built an FM-transmitter with the least amount of general-purpose parts ever.

To see how I arrived at that conclusion, you'll have to know how the clock system of an ATTiny45 works. Just like almost every other AVR, this little 8-pin microcontroller has the possibility of generating an 8MHz clock signal using an built-in oscillator. This is quite handy when the AVR you're using has only a few pins; using the internal oscillator, you free the two pins you normally connect a crystal to.

The internal RC oscillator, however, has two major disadvantages. When used directly, the maximum clock limits the speed of the AVR: while most parts can run at 16 or 20 MHz, the fixed 8MHz clock means an AVR is only half as quick as its maximum speed, at most. The second problem is that the clock oscillator isn't precise: its speed can vary tens of percents between parts. Atmel solved these problems in the later parts, like the ATTiny45: First, they added a calibration register to tune the frequency of the RC-oscillator. They also pre-calibrated the RC-clock to 8MHz and stored the calibration value to reach that inside the AVR. The performance problem (and other problems, like slow PWM-speeds) they solved by adding a PLL to the AVR: this device takes the 8MHz from the RC-oscillator and multiplies it by eight, spitting out a 64MHz signal. This is then divided by 4, giving a nice 16MHz signal to run your AVR on.

With all that, the AVR (which has a maximum clock speed of 24MHz) can run on 16MHz from its internal clock. Sometimes, even the 16MHz is not enough, and some time ago, I found myself in a position where I could use that extra 4MHz. While browsing the datasheet, I found a way to do that: by writing the right value in the RC oscillator calibration register, you could make it run up to 50-100% quicker! So, I decided to take whatever was in the register, add 50% and use that as the new calibration value. All of a sudden, my AVR was running on a nice 24MHz.

Then I started thinking... if I overclock the AVR to 24MHz, the PLL runs at 4 times that frequency: 96MHz. Hmmm, that's smack dab in the middle of the FM radio band. FM means Frequency Modulation, which basically means the signal an FM-transmitter spits out is a carrier frequency, say 96MHz. That frequency is basically lowered and heightened a bit when the modulated audio signal is lower or higher. Hmm, using the calibration register, I should be able to do that...

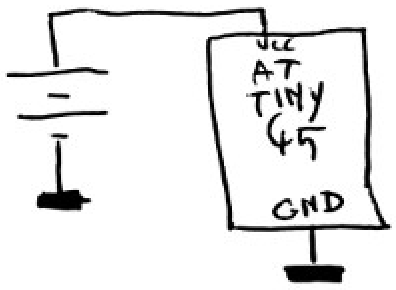
1 Next »

SpritesMods.com

AVR-based FM-transmitter - The transmitter

The transmitter

So, without further ado, here is the schematic of the transmitter I built:



Yes, you've seen that right. All that's there is an AVR powered by a battery. The firmware does all the work. It overclocks the CPU to 24MHz, making the PLL run at 96MHz. It then starts a simple music player. Instead of routing the output of the player to a pin which is connected to a speaker, I feed it into a routine which varies the 96MHz PLL frequency by just a little bit, thereby frequency modulating the signal.

The music player, by the way, is a small routine I made ages ago. It takes a music file made by MONOTONE, a music program for composing PC Speaker tunes on an IBM PC-XT. Because the sound hardware it's made for is so simple, it was quite easy to make a player for the AVR for it.

For a whimsy schematic like that, a whimsy build should work:

You're looking at the programmed AVR, with the power supply wires lengthened a bit. They're connected to two button cells, which are connected together by a bit of ducktape. (And yes, the two buttons plus the AVR add up to three components... if you want just two components, use a single LiIon cell.)

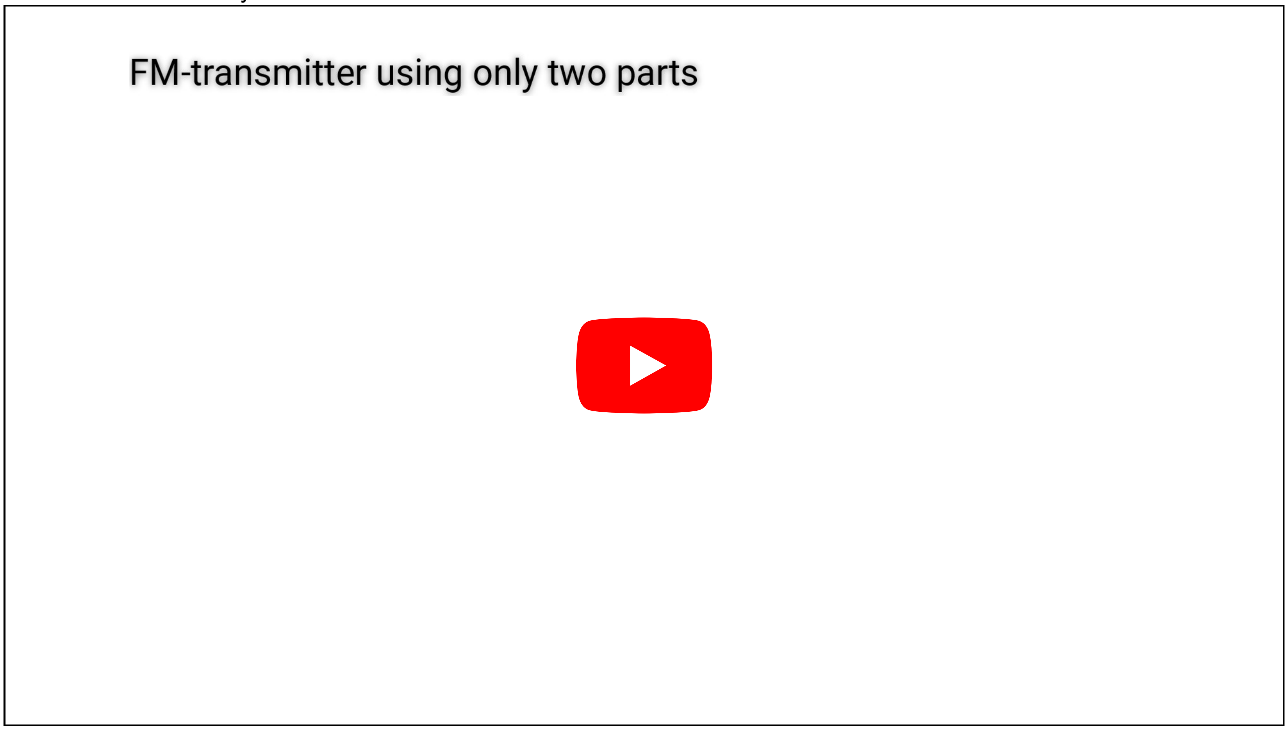
« Prev 2 Next »

SpritesMods.com

AVR-based FM-transmitter - Conclusion

Conclusion

Does it work? See for yourself!



The song that plays wasn't tracked by me, by the way. It came in the Monotone zip as an example song, and according to the documentation, it was made by someone called virt.

As usual, the firmware is released under the GPLv3 and you can download it here. Be advised that this is a highly experimental transmitter: it'll probably output noise on restricted bands. When built in the way I used it, I'm fairly sure that due to the limited range, you shouldn't get into trouble, but please don't try amplifying the signal.

« Prev 3

37 comments

ItsMagic wrote at 14 Mar 2023, 17.29:

Another fun hack: I've sizably extended range of mentioned thing by stumbling on GPS simulator, getting idea C/A spread sequence and correlation based receive is fun thing to try. I've used quite literal XOR of sequence output and signal, each big bit made of 1023 chips just like GPS does, except I dont use PSK link, I dont do +/-1 math and rather transferred all that to simplified, binary, 0/1 representation, correlation now mere XOR of 1023-bit long bit sequence, transmitting is as simple as sending 1023-bit sequence itself for "1" and inverse of it for "0" (doing honest XOR from gpecs would also do). Nobody promised me such math transformation is workable. Nobody promised me it works for OOK links instead of PSK. And yet... "Searching... C/A LOCK" said firmware. And yes, this magic works. Now that weird thing can punch bit perfect message through whole house (at much reduced bitrate though). Guess it counts as hack. On side note, this spread-spectrum stunt does suffers from TX and RX clock skew, GPS uses very tight lock of clocks using VCTCXO, but I dont have such luxury. This said I did fixup logic on correlator level to recover, if it unlocks it attempts few left/right shifts to test idea its a bit clock skew and this usually relocks back. That's what one gets for getting curious how far you can go on given hardware...

ItsMagic wrote at 30 Dec 2021, 13.54:

Hi! I've got inspired by this idea and... I had STM32F100. Small, neat QFP-48, on board I etched myself using "toner xfer" for other goals. Then I had some fun with "bare metal coding" on this thing. No libs, no crap, just me and this thing. Zero of foreign code. That's how you learn your hardware for the real. Somewhere along the lines I got idea

STM23 got MCO pin, that is, system clock output. And I even been smart enough to route that out of QFP on my DIY board. Gets interesting, right? I've got curious if MCU alone can do some RF, just like this project showcases. STM32F100 max clock is 24MHz. It sounds a bit too low for interesting bands. However, it got PLL. I've been smart enough to solder 12MHz xtal as well. So I've had what I really need to get it going. But I still want e.g. 27MHz band. 27MHz band is ISM, thus perfectly suitable for what looks like experiment, right? But this still out of spec. However, I got idea I can use AHB prescaler to put whole chip to /2 so it would be going 13.5MHz. The only question is whether PLL is going to lock out of spec. Some strange baremetal code... hmm... timeout? Oh, someone stupid forgot to turn on XTAL (HSE in STM32 names). And now... "pll locked"? Whoa. Only tiny part of chip, PLL itself runs at 27MHz, rest /2 by prescaler. So it is within specs. And yet I got really interesting clock to spit on MCO pin. It made as 12/4*9, so, in this setup, e.g. more classic STMF103 would not do as it lacks reasonable pre-divider. On other hand my thing hit ISM band. But I don't really want FM. I've designed 27MHz superregen similar to what old RC models control uses, but it explicitly tarteted "data link" using LM358-based amplifier and data slicer very similar to what jap.hu/ electronic/rf/el9805.pdf does except it used 27MHz superreg as 1st stage. Now what? Let's do OOK and spit some bits into the air! There're two issues. First, 27MHz needs BIG antenna to be efficient. Quarterwave is about 2.5 meters. I've used reduced coiled version soldered to MCO. Then MCO pin isn't exactly strong transmitter. It just spits 27MHz clock. Then receiver isn't super-good and 27MHz band is very noisy. But I can pick up my bits few meter away. Isn't it fun? So MCU along turned into 27MHz OOK transmitter, well, the only thing added is antenna. This design not relies on harmonics, main carrier hits ISM band instead. Thanks for inspiration!

rabin niroula wrote at 3 May 2020, 3.22:

Hey can you please provide me with the compiled hex file? my email : rabinniroula99@gmail.com

rabin niroula wrote at 3 May 2020, 3.21:

Hey can you please provide me with the compiled hex file?

ExileMage wrote at 7 Jan 2017, 0.44:

If I enable the CKOUT fuse then connect that to a coil of copper wire to use as an antenna, will it amplify the signal/ range?

Bender wrote at 30 Mar 2016, 8.58:

Hi Sprite, thank you for inspiration!

Rasa wrote at 13 Jan 2016, 17.25:

Hi Sprite, and thank you for sharing such inspiring projects with us . I was really excited to test this on my attiny but because I'm rather new to this world of micro controllers and I was confused by the firmware you've shared .Could you please describe how to upload it to the chip?

ravishankar wrote at 6 Jul 2015, 19.00:

excellent superb coding..hatsoff...

paranoja wrote at 29 May 2015, 10.02:

Great idea, and good coding. Mine attiny was tx-ing at 108 MHZ, found it using RTLSDR. thinkin about making a script which picks up attiny tx and translates it through SDR, maybe packing some usefull info in music :)

Ghaith wrote at 24 Mar 2015, 4.30:

thank you for this interesting idea :) i'd like to ask you two questions please , the first one : can i use ATTINY 11 or ATTINY 13 instead of ATTINY 45 ? The second question : would you like helping me and upload the (.hex) or thr written code file for me and i'll be thankful for you generosity

Q. wrote at 29 Nov 2014, 17.10:

You are cruel - but you conveniently blame &quot;virt&quot;. Your fan base will never forget you, or let you down or even give you up.

Brett wrote at 5 Jun 2014, 21.43:

Ignore that :) Got it.

Brett wrote at 5 Jun 2014, 20.18:

Great project. I might be missing something but when I try compile it is asking for &quot;io.h&quot; (#include from music.c file? Where would I get that? Thanks

Dimitris Zervas wrote at 8 Dec 2013, 0.38:

How can we amplify it (just a tiny bit) to make it work for 0.5-1m?

fogg wrote at 6 Nov 2013, 19.50:

I tried it and it worked - somewhat. I only get 18 MHz CPU clock, which is enough however, to receive the music around 90 MHz. I guess it depends on the original OCCAL value. In any case, when you enable the fuse CKOUT (i.e. 0xb1 instead of 0xf1 for the low fuse), you can get the clock signal on pin 3 (usually PB4). When you attach a small patch cable to it, you get much more poser for your bucks ;-).

Someone wrote at 9 Jul 2013, 0.39:

Which fuses does it need?

atmelfreak wrote at 7 Mar 2013, 18.18:

can you please upload the completed .hex file?

Sprite_tm wrote at 21 Feb 2013, 7.14:

EschatologicalEngineer: A drum beat doesn't sound right... if you copies my code, you should get about the same sound as in the video on this page.

EschatologicalEngineer wrote at 21 Feb 2013, 5.42:

Greetings Sprite, I have been following your projects for awhile and just decided to go ahead and give this one a shot for myself. I am really impressed by the simplicity of this brilliant project. This is a prime example of the potent data contained within Datasheets. I particularly enjoy reading Atmel datasheets, maybe something is wrong with me, but it has played a huge role in my didactic effort to acquire mad K-rad circuit skillz. I just wanted to check and make sure that my excitement was not due to the placebo of random interference or a stray signal sneaking inbetween those stations circa 96MHz.. Can you verify if the resulting embedded monotone music should sound like a drum beat, sort of like a cadence? I was able to pick it up very well directly above the ATTiny45 as it overpowered the other station signals, but when the receiver was moved away from the chip, the cadence was no longer audible and the stations dominated the frequency. Thank you sprite, for all of your contributions. This project has been an inspiration to me and I am glad that I gave it a shot today. I do hope that what I did hear was indeed evidence of an modulated overclocked frequency. Keep up the good work and Stay Fluxy!! -EschatologicalEngineer-KS

resistor wrote at 22 Dec 2012, 14.20:

my proplem: Build started 22.12.2012 at 15:16:28 avr-gcc -mmcu=attiny45 -Wall -gdwarf-2 -Os -std=gnu99 -funsigned-char -funsigned-bitfields -fpack-struct -fshort-enums -MD -MP -MT main_pc.o -MF dep/main_pc.o.d -c

../../music/main_pc.c In file included from ../../music/main_pc.c:24:0: c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:61:33: error: expected ')' before 'bits' c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:62:2: error: expected ';' before 'void' c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:96:37: error: expected declaration specifiers or '...' before 'uint_32' c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:96:51: error: expected declaration specifiers or '...' before 'uint_32' c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:96:65: error: expected declaration specifiers or '...' before 'uint_32' c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:97:57: error: expected declaration specifiers or '...' before 'uint_32' ../../music/main_pc.c: In function 'main': ../../music/main_pc.c:42:5: error: 'ao_device' undeclared (first use in this function) ../../music/main_pc.c:42:5: note: each undeclared identifier is reported only once for each function it appears in ../../music/main_pc.c:42:16: error: 'ao' undeclared (first use in this function) ../../music/main_pc.c:43:5: error: 'ao_sample_format' undeclared (first use in this function) ../../music/main_pc.c:43:22: error: expected ';' before 'form' ../../music/main_pc.c:46:5: warning: implicit declaration of function 'ao_default_driver_id' ../../music/main_pc.c:47:5: error: 'form' undeclared (first use in this function) ../../music/main_pc.c:50:22: error: 'AO_FMT_NATIVE' undeclared (first use in this function) ../../music/main_pc.c:54:5: warning: implicit declaration of function 'ao_open_live' ../../music/main_pc.c:56:2: warning: too few arguments for format ../../music/main_pc.c:78:2: error: too many arguments to function 'ao_play' c:programmeatmelavr toolsavr toolchainbin../lib/gcc/avr/4.5.1/../../../../avr/include/ao/ao.h:97:6: note: declared here make: *** [main_pc.o] Fehler 1 Build failed with 13 errors and 3 warnings...

Sprite_tm wrote at 6 Dec 2012, 14.55:

atmelfreak: Thank you for that incredibly detailed symptom report. I, however, would like to suggest that if you think you need an editor to use this, you may be better off trying some simpler projects first.

atmelfreak wrote at 6 Dec 2012, 14.08:

it doesn't work. What for a programing editor i have to use?

Sprite_tm wrote at 22 Oct 2012, 8.11:

Favner: I don't see why not, to my knowledge the 85 is just a 45 with extra memory.

Favner wrote at 17 Oct 2012, 7.01:

Hi, Can I use the ATtiny85 instead if the ATtiny45 and get the same results? Will it the overclocking wirk? Thanks.

Sprite_tm wrote at 19 Mar 2012, 8.07:

John: I\'d think the signal strength should be higher when you use 2 coin cells. I haven\'t done any tests on battery longevity; could take a day to drain them, could take months. You want the 20PU version, the 20 stands for the maximum speed and this application even drives the speed a bit over that.

John wrote at 19 Mar 2012, 0.00:

I\'m new to avr programming. This has inspired me to make a bunch of these and install them in my friends\' cars (and change all of their radio presets to the station, bwahahaha) You mention using only 1 coin cell vs. 2. Is there a difference in signal strength? Or just how long it will last? Also, while shopping for the attiny45 I\'ve come across two types, the 10PU vs. 20PU. Which would you recommend for this project?

Andrè wrote at 30 Jan 2012, 20.09:

About getting signal out via timer1 - since signal is frequency divided, it should be square with strong 3rd harmonic, so tuning PLL to 66 Mhz will give 33Mhz timer1 output with 3rd harmonics of it radiating at 99Mhz - quite at FM range. Even though 3rd harmonic is ~5dB weaker, getting signal out to a pin can end up with radiated power even greater than from PLL itself.

Niek wrote at 28 Jan 2012, 16.19:

NOES I GOT RICK-ROLLED BY AN AVR!

vitaliy wrote at 28 Jan 2012, 2.50:

very, very cool!!! =)

Sprite_tm wrote at 27 Jan 2012, 14.11:

feudor: Afaik there\'s no way to output the 96MHz PLL clock anywhere. You can feed it into timer1 and set the divisor to the lowest value, but you\'d still only have a 48MHz output.

feudor wrote at 27 Jan 2012, 9.25:

Is it really impossible to output the signal on a pin? Apparently, timer 1 can be clocked from the pll. So with a clever choice of the timer registers, like a upper limit of 1. How fast can we toggle a pin ?

Sprite_tm wrote at 27 Jan 2012, 9.01:

plaes: Where? The 96MHz doesn\'t really come out of any pin. It\'s probably possible to make the power supply wires longer to get a bit better reach, though. hboy: The calibration register itself is RAM; the factory calibration setting AVR set is copied there every startup. You can then change it without wearing out the cells.

plaes wrote at 27 Jan 2012, 8.38:

Have you considered adding a bigger antenna?

hboy wrote at 26 Jan 2012, 22.55:

That\'s such a sweet idea, thanks for the inspiration! But how long will the calibration register cells last?

Sprite_tm wrote at 25 Jan 2012, 23.42:

FYI: I just ninja-editted the article. It used to say that I ran the clock at 20MHz, which multiplied by 4 gives us the FM-frequency of 100MHZ... which is obviously wrong. I just fixed the code to overclock the AVR to 24MHz, and now you can indeed receive it at 96MHz on an FM-receiver. Why the previous 20MHz worked, I don\'t know, but I guess the radio picked up the fifth harmonic of the (square-wave-ish) 20MHz base clock.

harrstein wrote at 25 Jan 2012, 23.12:

rickrolled :D

bolt wrote at 25 Jan 2012, 22.33:

Here I was, about to go to bed, and you have to come and show me this. No sleep tonight. Need to make this. Damn you :)

Leave a comment:

Your name:

What does this picture say?

Your comment:

Comment