Generating MSK144 directly for Beacons and Test Sources.

Andy Talbot G4JNT
December 2016

Overview

MSK144 is a high speed data mode introduced into WSJT-X to replace FSK441 for meteor scatter (MS) and other paths where short-term burst openings are likely to occur. Unlike all the other modes within the WSJT suite, MSK144 does not rely upon pure frequency shift keying to convey information symbols, but on a more efficient coherent modulation type called Minimum Shift Keying.

Messages sent using MSK144 utilise the same compressed format as most of the other WSJT type messages, ie two callsigns and a report or locator; callsign plus CQ or other messages, or 13 characters of plain text. WSJT-X also introduces one or two other message types. The message is compressed down to 72 source bits to which an 8 bit checksum is appended to (virtually) eliminate false decodes. The 80 resulting source bits are then expanded using a Low Density Parity Code (LDPC) to 144 one-bit channel symbols before modulating an audio tone at 2000 baud for upconversion in an SSB transmitter. 144 symbols at 2000 baud means the entire message is sent in a burst just 72 milliseconds long. Bursts are repeated end-to-end for fixed Tx/Rx durations of 5 , 10, 15 or 30 seconds each. For reception, just one complete burst (which can have a number of symbols in error) is all that is necessary for a complete decode.

Modulation Format

MSK is a coherent mode like PSK with all the attendant advantages in S/N performance that coherency offers, but can be transmitted with a constant envelope through non linear transmitters without frequency spreading. MSK can also be looked on as Frequency Shift Keying where the shift is *exactly* half the symbol or baud rate. Note the emphasis on exactly; close to it, or nearly the same, is not enough. Shift must be coherently related to symbols; or at least, coherently over the duration of a transmission burst.

Using a Numerically Controlled Oscillator (NCO) to generate an audio tone or RF carrier, this shift can be applied directly to the increment added to the accumulator term. One way to do this is to alternately programme the NCO with the wanted tone frequency plus or minus a quarter the baud rate, or +/- 500Hz for MSK144. Within the WSJT-X suite this method is applied to an audio tone of 1500Hz. The modulated waveform then covers the frequency range roughly 300Hz to 2.7kHz and is designed to fit though an SSB filter.

Alternative Ways to Generate a MSK144 Signal

If the '0' and '1' channel symbols (or bits) can be generated separately and stored, they can then be used in a stand-alone controller to reprogramme a DDS chip in real time, sending codes for F_{RF} + 500Hz and F_{RF} - 500Hz at 2000 times per second based on the pre-stored message symbols.

Another solution is to implement an audio DDS inside the PIC that generates an I/Q quadrature audio stream for direct upconversion. If negative frequencies are requested, (twos complement values sent to the frequency register), the I/Q polarity swaps giving the opposite sideband . By reprogramming the NCO to generate plus or minus 500Hz then sending the resulting output to a direct quadrature upconverter, MSK modulation at RF can be generated at any frequency the converter will work at.

Generating The Raw Data

The first step is to generate the 144 symbols for any given message. Fortunately there is a utility *MSK144CODE.EXE* contained within the WSJT-X suite (look for it in the .BIN folder of the installation) to do just this. On its own, output is just sent to the user screen but by using redirection the output can be sent to a .TXT file. To do this issue the command :

`MSK144CODE "message info" > textfile.txt` replacing the quoted message info and file name as you want.

The additional (Windows) utility *GENMSK144* , contained in [1], provides an easier to use 'wrapper' for this software. *MSK144CODE* is run as a 'shell' command, redirecting the output to a text file which it then parses to extract the numerical data. The resulting symbols are then formatted and stored in a manner suitable for direct import to PIC assembly files. Operation of *GENMSK144* should be self explanatory. Ensure it sits in the same folder as *MSK144CODE* and that the four .DLL files the latter calls up are also present.

Typical output from *MSK144CODE* looks like this :

```
C:\msk144code "g4jnt MSK144" > q.txt

[contents of q.txt]
   Message          Decoded          Err? Type
------------------------------------------------------------------------
 1.  G4JNT MSK144         G4JNT MSK144          6: Free text

Channel symbols (72 per line):
110000111110001100010101101111100000110100000011100011111100001110110101
001101011010111100001100110000110001010000100010111100101011000000011001
```

If using *GENMSK144*, the screen will look something like this :

```
C:\genmsk144

Generate MSK144 PIC include file from  message
Calls MSK144CODE.EXE

Message              g4jnt msk144

 ;MSK144 symbols stored 8 per byte, MSB first.   Message   G4JNT MSK144

   de  b'11000011', b'11100011', b'00010101', b'10111110'
   de  b'00001101', b'00000011', b'10001111', b'11000011'
   de  b'10110101', b'00110101', b'10101111', b'00001100'
   de  b'11000011', b'00010100', b'00100010', b'11110010'
   de  b'10110000', b'00011001'
Symbols stored in MSK144SYMBS.INC

Any key to exit ...
```

Controlling a DDS Chip

A DDS chip can generate RF carrying MSK144 modulation by reprogramming it in real time at the 2000Hz symbol rate. Apart from any initial DDS setup codes, two sets of data need to be stored in the controller: The register value that needs to be programmed in for the actual RF centre frequency for a given DDS clock, referred to as N_{DDS} , and the value for generating 500Hz using that same clock, referred to as Delta. For example, using a 32 bit DDS with a 200MHz clock to generate at a centre frequency of 50.1MHz :

N_{DDS} = 50.1MHz / 200MHz * 2^{32} = 0x4020C49C

Delta = 500Hz / 200MHz * 2^{32} = 0x29F1

We need to be aware that due to 32 bit integer rounding, the actual frequency shift may not be exactly 500Hz. For the example here it works out at 499.98Hz. However, the resulting error of 0.02Hz corresponds to just 0.5degrees of phase shift over an entire 72ms burst, so its effect will go completely unnoticed. A 48bit DDS removes even this small error.

The controller code needs to generate a timer based interrupt at 2000Hz. Every interrupt period, the next symbol is extracted from the stored list generated above, then , if it is a '1' the value Delta is added to N_{DDS} If the symbol is a '0' Delta has to be subtracted. The resulting modified value of N is then sent to the DDS. Ensure the addition and subtraction routines take an equal number of clock cycles to avoid any timing asymmetry errors.

The controller also needs to maintain a symbol count from 0 to 143 to ensure the right bits are extracted in turn, and the pointer is reset at the end. It also needs to maintain a frame counter , if data is to be sent in bursts and not continuously, in order to control transmission and idle periods. Synchronising to an external 5 or 10 second marker signal is not necessary as the WSJT-X software receives this mode virtually independent of timing. A seconds counter, incremented every 2000 symbols and reset after a defined repeat period is useful for scheduling timed or sequenced message generation.

PIC assembly code *MSK144_9852.ASM* to be found in [1] contains a complete assembler listing for generating MSK144 from an AD9852 DDS chip using an 16F628A PIC controller at 10MHz. This is a relatively old 48 bit DDS with a 300MHz maximum clock frequency, but modifying the code to suit other devices should be straightforward. The timing is based on a 10MHz crystal, making it hardware compatible with other WSJT generating software driving this DDS chip.

Self Contained I/Q Audio DDS

A PIC microcontroller can be used with dual external D/A converters as a DDS generating at audio frequencies. The circuit diagram is shown in Figure 2, and more details can be found at [2] for a 24 bit DDS clocked (sampled) at a few tens of kHz. The reference gives hardware details for a suitable source ideally suited to this application. By selecting a suitable sampling rate for the audio, a 2000Hz timing epoch for extracting MSK144 symbols can also be generated and the raw data can be formed the same way as for the separate DDS described above. This time only the N_{DDS} value for 500Hz at the chosen sampling rate selected need be stored. At each sampling interval N_{DDS} is either added to, or subtracted from, the accumulator depending on the state of the last recalled symbol value.

There are a number of criteria to be met when asking the controller to do these two tasks, generating an audio waveform and handling the MSK144 timing – all in real time. One in particular, the choice of processor clock frequency needs to be made with care.

An interrupt is generated at the audio sampling frequency which must be an exact multiple of 2000Hz to allow the symbol timing to be generated inside the timer interrupt routine. Ideally the sampling should be at a frequency that allows 500Hz be generated from the 24 bit DDS with no rounding error, but in practice, any resulting error will be minute, probably less so than the example given above.

Now, instead of adding or subtracting 500Hz, depending on the symbol, the N_{DDS} value for 500Hz is either used as is for a '1' or negated (twos complemented) for a '0'.

To generate a timer interrupt at a multiple of 2000Hz dictates that only certain frequencies can be used for the processor clock. A PIC needs a crystal running at four times its processor clock; a prescalar of the clock before hitting the interrupt counter is useful and a value of 2 was selected. So to get a timer interrupt at a multiple of 2kHz needs an oscillator crystal that is a multiple of 2kHz * 2 (prescale) * 4 or a multiple of 16kHz and has to be one that will give a suitable sampling frequency for 500Hz generation. 10MHz as used for the separate DDS example does not work, but 16MHz or 12.8MHz or 12.288MHz which are all off the shelf crystal frequencies are allowed. The latter was the one chosen as a suitable crystal was to hand. The sampling rate also needs to be low enough that all the interrupt code for the most complex set of instructions is completely finished before the next one occurs. In the example here a 16kHz sampling rate meets all criteria, but 32kHz was just a bit too fast. Had a 16MHz crystal been used, 32kHz sampling would have been possible with its attendant relaxation in anti-alias filtering.

PIC assembler code *MSK144_IQ.ASM* to be found in [1] contains a complete listing for using a 16F870 with dual R-2R ladder D/A converters. Constants for different clock frequencies are all listed at the start. To assist practical use, two links are used to set up. Connecting port A0 to ground swaps the polarity of the I/Q. This saves having to disconnect and swap over the I and Q drive signals to the upconverter when it is discovered the wrong sideband is being generated. Connecting A1 to ground generates a single tone, this is the low tone corresponding to a data '0' and a continuous DDS frequency of -500Hz. The correct I/Q polarity can be checked by monitoring the resulting shift on a receiver.

A Tx/ Idle time period is also included. Set using a constant shown at the start of the .ASM file, this is the repeat time for NFRAMES of data. Choosing *NFRAMES* equal to 69 and *REPEAT* at 10 seconds gives a 5 second Tx, 5 second off period that matches the WSJT-X shortest cycle period for this mode. Port A2 provides a logic signal that goes high during tone generation and low during the idle period when no tone is there. This can be used to control a transmitter PA stage to reduce power consumption during off periods.

Port A3 of the PIC provides a strobe signal for the duration on the interrupt. It is set on entry and cleared on exit from the interrupt routine so its width varies depending upon how many tasks have been performed each time. Monitoring this strobe signal will show if a particular sampling rate/ interrupt rate is too much for the commands executed, and allows an optimum sampling rate to be selected.

As the processor clock controls both timing of the MSK144 waveform and audio frequency, clock should be adjusted to get as close to 10MHz as possible. A trimmer capacitor added on the OSC-In pin helps facilitate this.

A Complete 1.3 / 2.3GHz MSK144 Beacon Source

The low frequency DDS modulator described produces an I/Q baseband signal that needs to be upconverted to an RF signal at the wanted frequency. To generate at 1.3GHz, the ADF8346 integrated quadrature upconverter was used since a PCB and demonstrator have previously been made and tested. [3] This chip will work over the frequency range of around 800MHz to 2.5GHz so will happily cover the 1.3 and 2.3GHz bands. The circuit diagram of the upconverter circuitry can be seen in Figure 3.

The AD8346 requires differential baseband drive, so opamp buffers were included to define DC voltage levels. Ideally, direct upconversion should include a DC path right through to the mixer to allow any true centre frequency carrier to pass. However, allowing a DC path is a recipe for leakage of unwanted signals when it is not properly nulled, and in many applications, particularly for MFSK and MSK generation, there is no low frequency baseband component present meaning capacitor coupling will suffice. For MSK144 the modulation is a plus/minus 500Hz term (or OQPSK depending on your view) which has an amplitude null at zero frequency.

The analogue circuitry used is a bit excessive in terms of component count and opamp chips as it was all made from modules that already existed. An integrated solution could no-doubt make use of differential line drivers, or other configurations to simplify the drive circuitry.

As an alternative, opamp buffers could be added with output networks presenting a 50Ω source for driving passive diode mixers. This allows the modulation to be generated at high frequencies with diode subharmonic mixers or DBMs. Suitable drive details can be found at [4]

To generate the RF, a ready made ADF4351 module was used – one of those popularised on Ebay – which can generate RF over the frequency band covered by the upconverter. A dedicated PIC controller allows link or switch selection of several pre-programmed frequencies. This can be seen piggy-backed on top of the synthesizer PCB.

References

[1] http://www.g4jnt.com/MSK144_PIC.zip

[2] http://www.g4jnt.com/PIC_DDS.pdf with accompanying files
http://www.g4jnt.com/PIC_DDS.zip

[3] '*Integrated Quadrature Up and Down Converters*' . RadCom Feb 2012, page 34.

[4] '*Diode mixer as a PSK modulator*' RadCom April 2016, page 26

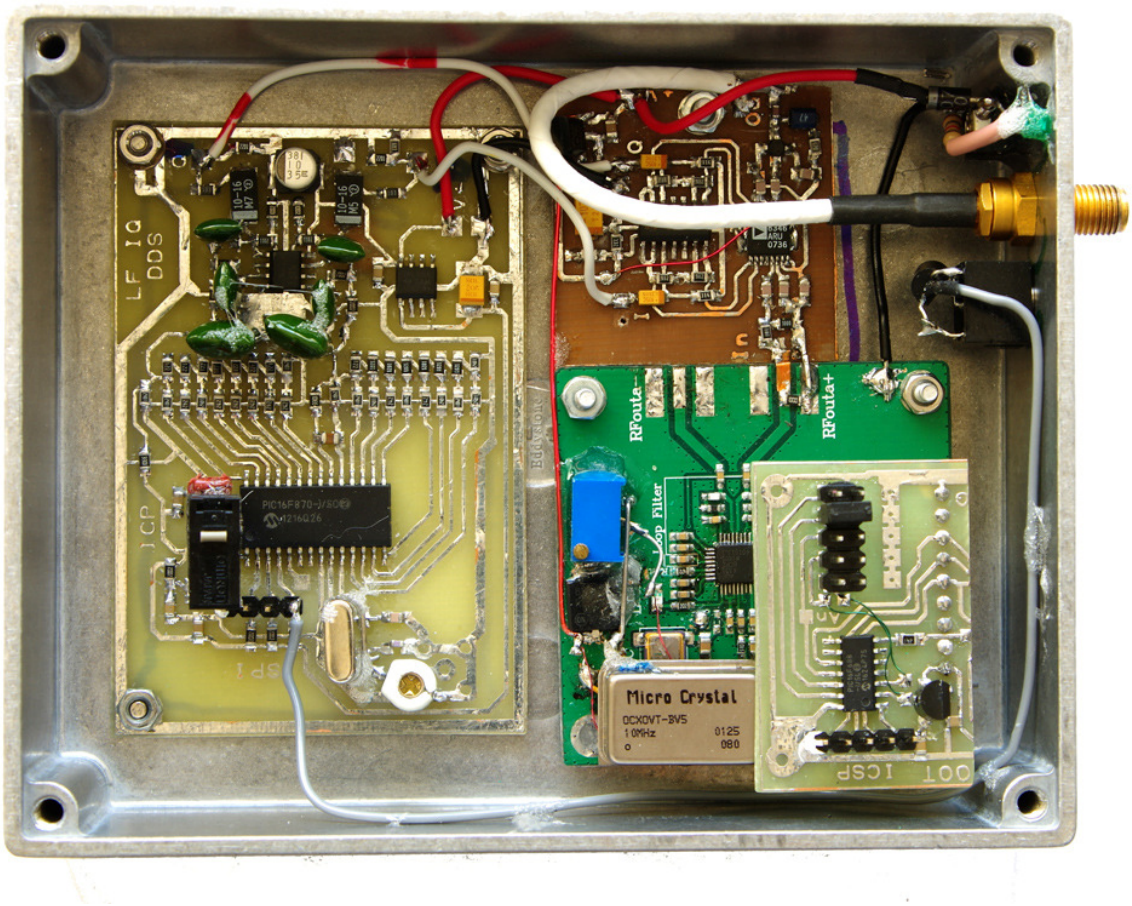

Figure 1 The complete MSK144 Source for 1.3 and 2.3GHz

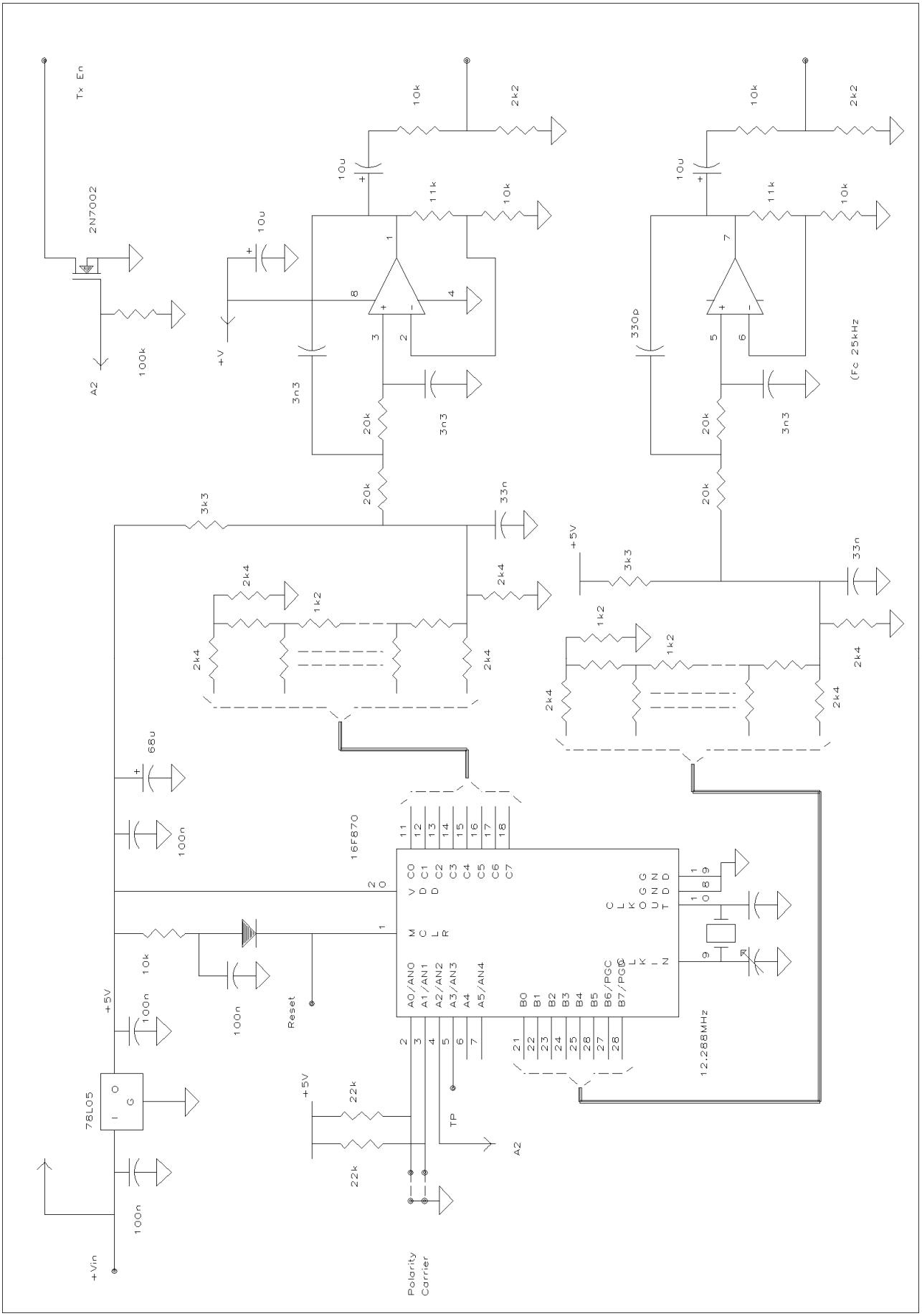

Figure 2 Circuit diagram of PIC DDS / modulator

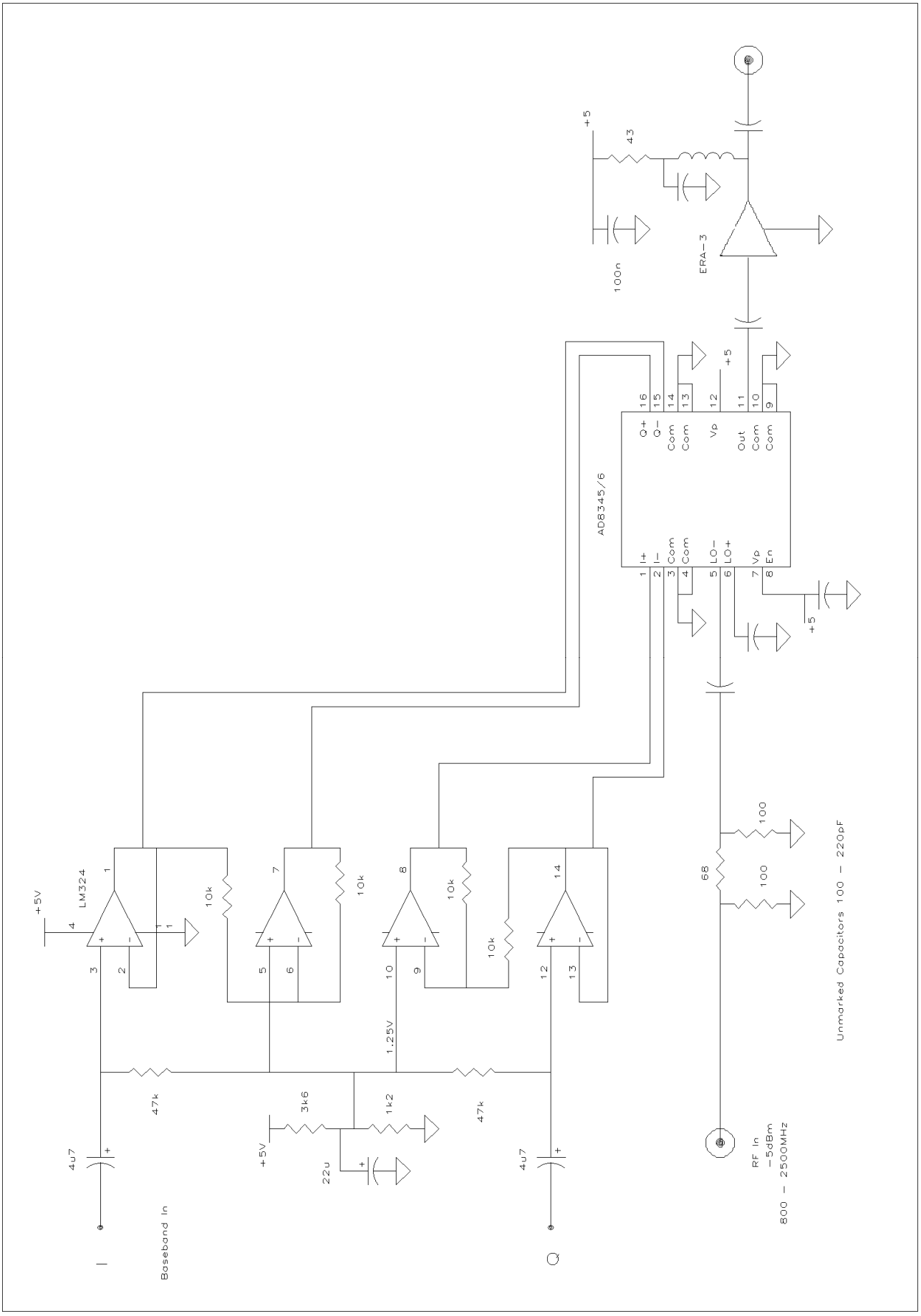

Figure 3 RF Quadrature Upconverter