Python Radio 18: DominoEX Mode

Simon Quellen Field

Simon Quellen Field

Follow

10 min read

·

Sep 5, 2024

Listen

Share

More

Perfect for weak signal communication.

Press enter or click to view image in full size

Image by the author

Another mode that is great for NVIS and weak signal work on the HF bands is DominoEX.

Like FSQ, it is a conversational mode that is easy to tune, tolerates frequency drift, has low bandwidth but high typing speed, works well for NVIS conditions, and is robust enough to fading and interference that it normally needs no forward error correction.

DominoEX (often just called Domino, since the previous versions such as DominoF are no longer in use) has 6 baud rates. The lowest, DominoEX4, runs at 3.90625 baud, uses 173 hertz of bandwidth, and manages as much as 25 words per minute. The other rates are DominoEX5, 8, 11, and 22 (the actual baud rates are 5.3833, 7.8125, 10.766, 15.625, and 21.533). The words per minute are respectively 31, 50, 70, 100, and 140.

There is also a DominoEX Micro mode that uses 2 baud, DominoEX44, and DominEX88 that were not in the original specification.

Like FSQ, DominoEX uses Incremental Frequency Keying, where symbols are defined by the difference between tones, rather than the tones themselves. This makes tuning easy (you can be 200 hertz off frequency and still decode the message) and allows it to tolerate frequency drift.

There are 18 tones, and the varicode tables are grouped into sets of three tones. The dominoex\_varicode.py module looks like this:

Dominoex\_varicode = [

# Primary alphabet

( 1,15, 9), ( 1,15,10), ( 1,15,11), ( 1,15,12), ( 1,15,13), ( 1,15,14), ( 1,15,15), ( 2, 8, 8),

( 2,12, 0), ( 2, 8, 9), ( 2, 8,10), ( 2, 8,11), ( 2, 8,12), ( 2,13, 0), ( 2, 8,13), ( 2, 8,14),

( 2, 8,15), ( 2, 9, 8), ( 2, 9, 9), ( 2, 9,10), ( 2, 9,11), ( 2, 9,12), ( 2, 9,13), ( 2, 9,14),

( 2, 9,15), ( 2,10, 8), ( 2,10, 9), ( 2,10,10), ( 2,10,11), ( 2,10,12), ( 2,10,13), ( 2,10,14),

( 0, 0, 0), ( 7,11, 0), ( 0, 8,14), ( 0,10,11), ( 0, 9,10), ( 0, 9, 9), ( 0, 8,15), ( 7,10, 0),

( 0, 8,12), ( 0, 8,11), ( 0, 9,13), ( 0, 8, 8), ( 2,11, 0), ( 7,14, 0), ( 7,13, 0), ( 0, 8, 9),

( 3,15, 0), ( 4,10, 0), ( 4,15, 0), ( 5, 9, 0), ( 6, 8, 0), ( 5,12, 0), ( 5,14, 0), ( 6,12, 0),

( 6,11, 0), ( 6,14, 0), ( 0, 8,10), ( 0, 8,13), ( 0,10, 8), ( 7,15, 0), ( 0, 9,15), ( 7,12, 0),

( 0, 9, 8), ( 3, 9, 0), ( 4,14, 0), ( 3,12, 0), ( 3,14, 0), ( 3, 8, 0), ( 4,12, 0), ( 5, 8, 0),

( 5,10, 0), ( 3,10, 0), ( 7, 8, 0), ( 6,10, 0), ( 4,11, 0), ( 4, 8, 0), ( 4,13, 0), ( 3,11, 0),

( 4, 9, 0), ( 6,15, 0), ( 3,13, 0), ( 2,15, 0), ( 2,14, 0), ( 5,11, 0), ( 6,13, 0), ( 5,13, 0),

( 5,15, 0), ( 6, 9, 0), ( 7, 9, 0), ( 0,10,14), ( 0,10, 9), ( 0,10,15), ( 0,10,10), ( 0, 9,12),

( 0, 9,11), ( 4, 0, 0), ( 1,11, 0), ( 0,12, 0), ( 0,11, 0), ( 1, 0, 0), ( 0,15, 0), ( 1, 9, 0),

( 0,10, 0), ( 5, 0, 0), ( 2,10, 0), ( 1,14, 0), ( 0, 9, 0), ( 0,14, 0), ( 6, 0, 0), ( 3, 0, 0),

( 1, 8, 0), ( 2, 8, 0), ( 7, 0, 0), ( 0, 8, 0), ( 2, 0, 0), ( 0,13, 0), ( 1,13, 0), ( 1,12, 0),

( 1,15, 0), ( 1,10, 0), ( 2, 9, 0), ( 0,10,12), ( 0, 9,14), ( 0,10,13), ( 0,11, 8), ( 2,10,15),

( 2,11, 8), ( 2,11, 9), ( 2,11,10), ( 2,11,11), ( 2,11,12), ( 2,11,13), ( 2,11,14), ( 2,11,15),

( 2,12, 8), ( 2,12, 9), ( 2,12,10), ( 2,12,11), ( 2,12,12), ( 2,12,13), ( 2,12,14), ( 2,12,15),

( 2,13, 8), ( 2,13, 9), ( 2,13,10), ( 2,13,11), ( 2,13,12), ( 2,13,13), ( 2,13,14), ( 2,13,15),

( 2,14, 8), ( 2,14, 9), ( 2,14,10), ( 2,14,11), ( 2,14,12), ( 2,14,13), ( 2,14,14), ( 2,14,15),

( 0,11, 9), ( 0,11,10), ( 0,11,11), ( 0,11,12), ( 0,11,13), ( 0,11,14), ( 0,11,15), ( 0,12, 8),

( 0,12, 9), ( 0,12,10), ( 0,12,11), ( 0,12,12), ( 0,12,13), ( 0,12,14), ( 0,12,15), ( 0,13, 8),

( 0,13, 9), ( 0,13,10), ( 0,13,11), ( 0,13,12), ( 0,13,13), ( 0,13,14), ( 0,13,15), ( 0,14, 8),

( 0,14, 9), ( 0,14,10), ( 0,14,11), ( 0,14,12), ( 0,14,13), ( 0,14,14), ( 0,14,15), ( 0,15, 8),

( 0,15, 9), ( 0,15,10), ( 0,15,11), ( 0,15,12), ( 0,15,13), ( 0,15,14), ( 0,15,15), ( 1, 8, 8),

( 1, 8, 9), ( 1, 8,10), ( 1, 8,11), ( 1, 8,12), ( 1, 8,13), ( 1, 8,14), ( 1, 8,15), ( 1, 9, 8),

( 1, 9, 9), ( 1, 9,10), ( 1, 9,11), ( 1, 9,12), ( 1, 9,13), ( 1, 9,14), ( 1, 9,15), ( 1,10, 8),

( 1,10, 9), ( 1,10,10), ( 1,10,11), ( 1,10,12), ( 1,10,13), ( 1,10,14), ( 1,10,15), ( 1,11, 8),

( 1,11, 9), ( 1,11,10), ( 1,11,11), ( 1,11,12), ( 1,11,13), ( 1,11,14), ( 1,11,15), ( 1,12, 8),

( 1,12, 9), ( 1,12,10), ( 1,12,11), ( 1,12,12), ( 1,12,13), ( 1,12,14), ( 1,12,15), ( 1,13, 8),

( 1,13, 9), ( 1,13,10), ( 1,13,11), ( 1,13,12), ( 1,13,13), ( 1,13,14), ( 1,13,15), ( 1,14, 8),

( 1,14, 9), ( 1,14,10), ( 1,14,11), ( 1,14,12), ( 1,14,13), ( 1,14,14), ( 1,14,15), ( 1,15, 8),

# Secondary alphabet

( 6,15, 9), ( 6,15,10), ( 6,15,11), ( 6,15,12), ( 6,15,13), ( 6,15,14), ( 6,15,15), ( 7, 8, 8),

( 4,10,12), ( 7, 8, 9), ( 7, 8,10), ( 7, 8,11), ( 7, 8,12), ( 4,10,13), ( 7, 8,13), ( 7, 8,14),

( 7, 8,15), ( 7, 9, 8), ( 7, 9, 9), ( 7, 9,10), ( 7, 9,11), ( 7, 9,12), ( 7, 9,13), ( 7, 9,14),

( 7, 9,15), ( 7,10, 8), ( 7,10, 9), ( 7,10,10), ( 7,10,11), ( 7,10,12), ( 7,10,13), ( 7,10,14),

( 3, 8, 8), ( 4,15,11), ( 5, 8,14), ( 5,10,11), ( 5, 9,10), ( 5, 9, 9), ( 5, 8,15), ( 4,15,10),

( 5, 8,12), ( 5, 8,11), ( 5, 9,13), ( 5, 8, 8), ( 4,10,11), ( 4,15,14), ( 4,15,13), ( 5, 8, 9),

( 4,11,15), ( 4,12,10), ( 4,12,15), ( 4,13, 9), ( 4,14, 8), ( 4,13,12), ( 4,13,14), ( 4,14,12),

( 4,14,11), ( 4,14,14), ( 5, 8,10), ( 5, 8,13), ( 5,10, 8), ( 4,15,15), ( 5, 9,15), ( 4,15,12),

( 5, 9, 8), ( 4,11, 9), ( 4,12,14), ( 4,11,12), ( 4,11,14), ( 4,11, 8), ( 4,12,12), ( 4,13, 8),

( 4,13,10), ( 4,11,10), ( 4,15, 8), ( 4,14,10), ( 4,12,11), ( 4,12, 8), ( 4,12,13), ( 4,11,11),

( 4,12, 9), ( 4,14,15), ( 4,11,13), ( 4,10,15), ( 4,10,14), ( 4,13,11), ( 4,14,13), ( 4,13,13),

( 4,13,15), ( 4,14, 9), ( 4,15, 9), ( 5,10,14), ( 5,10, 9), ( 5,10,15), ( 5,10,10), ( 5, 9,12),

( 5, 9,11), ( 3, 8,12), ( 4, 9,11), ( 4, 8,12), ( 4, 8,11), ( 3, 8, 9), ( 4, 8,15), ( 4, 9, 9),

( 4, 8,10), ( 3, 8,13), ( 4,10,10), ( 4, 9,14), ( 4, 8, 9), ( 4, 8,14), ( 3, 8,14), ( 3, 8,11),

( 4, 9, 8), ( 4,10, 8), ( 3, 8,15), ( 4, 8, 8), ( 3, 8,10), ( 4, 8,13), ( 4, 9,13), ( 4, 9,12),

( 4, 9,15), ( 4, 9,10), ( 4,10, 9), ( 5,10,12), ( 5, 9,14), ( 5,10,12), ( 5,11, 8), ( 7,10,15),

( 7,11, 8), ( 7,11, 9), ( 7,11,10), ( 7,11,11), ( 7,11,12), ( 7,11,13), ( 7,11,14), ( 7,11,15),

( 7,12, 8), ( 7,12, 9), ( 7,12,10), ( 7,12,11), ( 7,12,12), ( 7,12,13), ( 7,12,14), ( 7,12,15),

( 7,13, 8), ( 7,13, 9), ( 7,13,10), ( 7,13,11), ( 7,13,12), ( 7,13,13), ( 7,13,14), ( 7,13,15),

( 7,14, 8), ( 7,14, 9), ( 7,14,10), ( 7,14,11), ( 7,14,12), ( 7,14,13), ( 7,14,14), ( 7,14,15),

( 5,11, 9), ( 5,11,10), ( 5,11,11), ( 5,11,12), ( 5,11,13), ( 5,11,14), ( 5,11,15), ( 5,12, 8),

( 5,12, 9), ( 5,12,10), ( 5,12,11), ( 5,12,12), ( 5,12,13), ( 5,12,14), ( 5,12,15), ( 5,13, 8),

( 5,13, 9), ( 5,13,10), ( 5,13,11), ( 5,13,12), ( 5,13,13), ( 5,13,14), ( 5,13,15), ( 5,14, 8),

( 5,14, 9), ( 5,14,10), ( 5,14,11), ( 5,14,12), ( 5,14,13), ( 5,14,14), ( 5,14,15), ( 5,15, 8),

( 5,15, 9), ( 5,15,10), ( 5,15,11), ( 5,15,12), ( 5,15,13), ( 5,15,14), ( 5,15,15), ( 6, 8, 8),

( 6, 8, 9), ( 6, 8,10), ( 6, 8,11), ( 6, 8,12), ( 6, 8,13), ( 6, 8,14), ( 6, 8,15), ( 6, 9, 8),

( 6, 9, 9), ( 6, 9,10), ( 6, 9,11), ( 6, 9,12), ( 6, 9,13), ( 6, 9,14), ( 6, 9,15), ( 6,10, 8),

( 6,10, 9), ( 6,10,10), ( 6,10,11), ( 6,10,12), ( 6,10,13), ( 6,10,14), ( 6,10,15), ( 6,11, 8),

( 6,11, 9), ( 6,11,10), ( 6,11,11), ( 6,11,12), ( 6,11,13), ( 6,11,14), ( 6,11,15), ( 6,12, 8),

( 6,12, 9), ( 6,12,10), ( 6,12,11), ( 6,12,12), ( 6,12,13), ( 6,12,14), ( 6,12,15), ( 6,13, 8),

( 6,13, 9), ( 6,13,10), ( 6,13,11), ( 6,13,12), ( 6,13,13), ( 6,13,14), ( 6,13,15), ( 6,14, 8),

( 6,14, 9), ( 6,14,10), ( 6,14,11), ( 6,14,12), ( 6,14,13), ( 6,14,14), ( 6,14,15), ( 6,15, 8),

]

The dominoex\_config.py module looks much like the ones from MFSK and FSQ:

From dominoex import DOMINOEX

From radio import Radio

From time import sleep\_ms, sleep

Class DominoEXConfig:

Def \_\_init\_\_(self, baud, frq, call, location):

Self.dds = Radio()

Self.is\_beacon = False

Self.beacon\_interval = 30.0

Self.message = ‘’

Self.usb\_offset = 1350.0

Self.num\_tones = 18

Self.incremental\_tone = 0.0

Self.all\_done = False

Self.r = DOMINOEX(self.send\_tone, self.report\_all\_done)

Self.set\_baud(baud)

Self.frequency = frq

Self.call = call

Self.location = location

Self.r.set\_frequency(frq)

Self.r.set\_call(call)

Self.r.set\_location(location)

Def get\_radio(self):

Return self.dss

Def send\_code(self):

Self.dds.send()

Def send\_tone(self, tone):

Self.incremental\_tone = (self.incremental\_tone + float(tone) + 2) % self.num\_tones

Self.f = int(int(self.frequency) + self.usb\_offset + (self.incremental\_tone + 0.5) \* self.tone\_spacing – self.bandwidth / 2.0)

Self.dds.set\_freq(0, self.f)

Self.dds.send()

Def report\_all\_done(self):

Print()

Print(“All done!”)

Self.all\_done = True

If self.is\_beacon:

Self.r.stop() # stop sending bits

# self.dds.off()

Sleep(float(self.beacon\_interval))

Self.dds.on()

Self.r.send\_code() # Repeat for a beacon

Else:

Self.r.stop() # stop sending bits

# self.dds.off()

Def set\_baud(self, b):

Self.baud = b

If self.baud == 2:

Self.spaced = 1

Self.sample\_rate = 8000.0

Self.symbol\_length = 4000.0

Elif self.baud == 4:

Self.spaced = 2

Self.sample\_rate = 8000.0

Self.symbol\_length = 2048.0

Elif self.baud == 5:

Self.spaced = 2

Self.sample\_rate = 11025.0

Self.symbol\_length = 2048.0

Elif self.baud == 8:

Self.spaced = 2

Self.sample\_rate = 8000.0

Self.symbol\_length = 1024.0

Elif self.baud == 11:

Self.spaced = 1

Self.sample\_rate = 11025.0

Self.symbol\_length = 1024.0

Elif self.baud == 16:

Self.spaced = 1

Self.sample\_rate = 8000.0

Self.symbol\_length = 512.0

Elif self.baud == 22:

Self.spaced = 1

Self.sample\_rate = 11025.0

Self.symbol\_length = 512.0

Elif self.baud == 44:

Self.spaced = 2

Self.sample\_rate = 11025.0

Self.symbol\_length = 256.0

Elif self.baud == 88:

Self.spaced = 1

Self.sample\_rate = 11025.0

Self.symbol\_length = 128.0

Self.r.set\_baud(self.sample\_rate / self.symbol\_length)

Self.r.set\_bit\_length(1000 / (self.sample\_rate / self.symbol\_length))

Self.tone\_spacing = self.sample\_rate \* self.spaced / self.symbol\_length

Self.bandwidth = self.num\_tones \* self.tone\_spacing

Def set\_message(self, msg):

Self.r.set\_message(chr(0) + “\r” + msg + “\r”)

Self.dds.on()

Self.r.send\_code()

Self.all\_done = False

Print(”Frequency:”, self.frequency)

Print(”Baud:”, self.baud)

Print(”Beacon?:”, self.is\_beacon)

Print(”Message:”, self.r.message)

Print()

Print(”Bandwidth:”, self.bandwidth)

Print(”Tone spacing:”, self.tone\_spacing)

Print(”Symbol length:”, self.symbol\_length)

Print(”Bit length:”, 1000 / (self.sample\_rate / self.symbol\_length))

Print(”Baud:”, self.sample\_rate / self.symbol\_length)

Def set\_beacon(self, onoff, interval):

Self.is\_beacon = onoff

Self.beacon\_interval = interval

Much of it is simply setting up the various baud rates.

The dominoex.py module handles the translation between letters and tones, using the by-now-familiar generator we still call “bit” although it is once again sending symbols:

From machine import Timer

From dominoex\_varicode import dominoex\_varicode

Class DOMINOEX:

Def \_\_init\_\_(self, send\_tone, report\_message\_end=None):

Self.send\_tone = send\_tone

Self.report\_message\_end = report\_message\_end

# self.set\_baud(10.766) # DOMINOEX 11

Self.set\_baud(2) # DOMINOEX MICRO

Self.frequency = “7104000”

Self.call = “N0CALL”

Self.location = “CM87xe”

Self.message = “{} {} “

Self.bit\_length = int(1000 / float(self.baud))

Self.timer = Timer()

Def set\_call(self, call):

Self.call = call

Def set\_baud(self, baud):

Self.baud = float(baud)

Def set\_bit\_length(self, len):

Self.bit\_length = int(len)

Def set\_frequency(self, frequency):

Self.frequency = frequency

Def set\_location(self, location):

Self.location = location

Def set\_message(self, message):

Self.message = message.format(self.call, self.location)

Def bit(self):

For letter in self.message:

Code = dominoex\_varicode[ord(letter)]

Count = 0

For tone in code:

If tone & 0x8 or count == 0:

Yield tone

Count += 1

Self.report\_message\_end()

Def stop(self):

Self.timer.deinit()

Def send\_code(self):

Self.gen = self.bit()

Self.timer.init(period=self.bit\_length, mode=Timer.PERIODIC, callback=self.bit\_finished)

Def send\_bit(self, unused):

Try:

Tone = next(self.gen)

Except StopIteration as tone:

Return self.report\_message\_end()

Self.send\_tone(tone)

Def bit\_finished(self, unused):

Self.send\_bit(True)

Our main.py module looks like this:

From dominoex\_config import DominoEXConfig

From time import sleep

Def main():

Dex = DominoEXConfig(4, 7040000, “AB6NY”, “CM87xe”)

While True:

Dex.set\_message(“{} Testing from {} using a Raspberry Pi Pico RP2040”)

Dex.send\_code()

While dex.all\_done == False:

Sleep(5)

Main()

We set up the baud rate (4 in this case), and the frequency, call, and location. Then we set up the message, and start sending.

The radio.py and SI5351.py modules are the same as before.

I did notice when using a baud rate of 11 (and even 8) that the first few characters in the message would often be garbled due to FLDIGI trying to be too smart about adjusting signal levels. Since the call sign was what was getting garbled (not a good thing) I added a few disposable characters in the beginning:

Dex.set\_message(” 🡪 {} Testing from {} using a Raspberry Pi Pico RP2040”)

The result looked like this:

Press enter or click to view image in full size

Image by the author

Interestingly, there was less of this at the faster baud rate of 22 (140 words per minute). A few errors, but for the most part, completely readable:

Press enter or click to view image in full size

Image by the author

At 44 baud, things got progressively worse, but it was so fast that it was easy to get the whole message by reading several lines:

Press enter or click to view image in full size

Image by the author

To computer users used to a baud being a bit instead of a symbol, 44 baud sounds slow. However, since each symbol is a character in DominoEX, this is 44 characters per second or 440 bits per second. Still pathetic compared to Wi-Fi speeds, but this is five dollars’ worth of hardware capable of bouncing over the horizon. And as a chat mode, 280 words per minute is faster than I can type anyway.

Dominoex

Amateur Radio