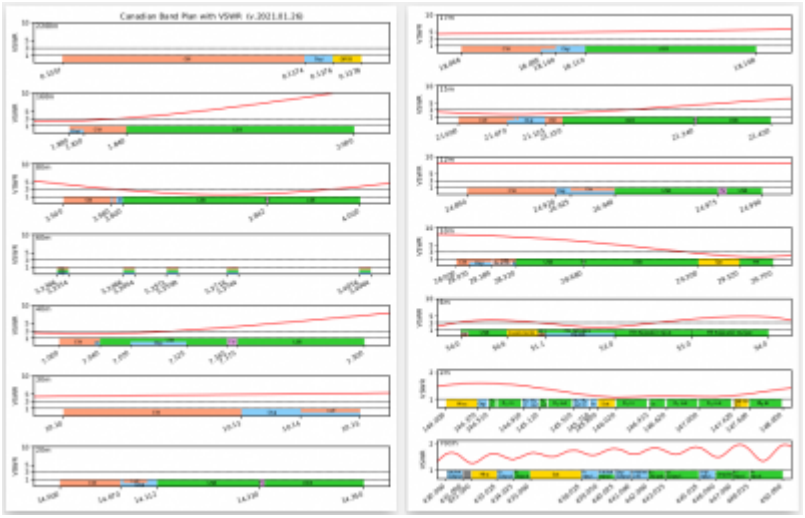
Band Plan With SWR

This python script does two things:



1. It reads all `.asd` output files from the Rig Expert that are placed in the same folder as this script and calculates the VSWR as a function of the frequency.
2. It recreates the band plan for all bands from 2200m to 70cm and adds VSWR graphs on top of it.

The HF band plan information was taken from
https://www.rac.ca/operating/bandplans/

The VHF and UHF band plan information is for British-Columbia and was taken from https://bcarcc.org/

Here's the output of my G5RV and GP9 so far:

- The top of the band plan graph corresponds to SWR = 1:1, where the first dotted horizontal line is.
- For HF, I also added a horizontal line at SWR = 3:1, which is where my internal tuner can reach.
- And the top of the graph is at SWR = 10:1, which is where most external tuners can reach.
- The red graph is the SWR curve. If it's not showing, it's because it's above 10:1

Update (Dec 8, 2024): I added the 630m band and updated the 60m band plan

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 29 16:40:41 2021

License:
   This script by Patrick Truchon <va7fi@rbox.me> is licensed
   under a Creative Commons Creative Commons Attribution-Share Alike 4.0
   Unported License.  <http://creativecommons.org/licenses/by-sa/4.0>.

   You are free to:
      * Run the scripts for any purpose.
      * Study and modify the scripts.
      * Copy the scripts to help others.
```

```
      * Improve the scripts, and release the improvements to the public, so
        that the whole community benefits.

   Provided that you:
     * Attribute the work to me by linking to
       <https://scarcs.ca/links/band_plan_with_swr>
     * Distribute any derivative work under the same license.

This script does two things:
   1) It reads all .asd output files from the Rig Expert that are placed
      in the same folder as this script and calculates the VSWR as a function
      of the frequency.
   2) It recreates the band plan for all bands from 2200m to 70cm and adds
      VSWR graphs on top of it.

For HF, the band plan information was taken from:
   https://www.rac.ca/operating/bandplans/

For VHF and UHF, the band plan information is for British-Columbia and
was taken from:
   https://bcarcc.org/

Versions
  * 2021.01.30
  * 2024.12.08 (added 630m and updated 60m band plan)
"""

import os
import json
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
from matplotlib.backends.backend_pdf import PdfPages

# The output will be a multipage PDF document:
pdf_pages = PdfPages('CanadianBandPlan.pdf')
TITLE = 'Canadian Band Plan with VSWR  (v.2024.12.08)'

#### Import SWR Data from Rig Expert .asd files ####
# Make a list of the .asd files in current folder and sort them by name.
PATH = r"./"
DIR = os.listdir(PATH)
FILES = []
for file in DIR:
    if file.endswith(".asd"):
        FILES = FILES + [file]
FILES.sort()

# Read FILES list and create lists for Frequencies, Resistance and Reactance:
FREQ = []
RESISTANCE = []
REACTANCE = []
```

```
for file in FILES:
    with open(file, "r", encoding="utf-8") as raw_file:
        raw = json.load(raw_file)
    SIZE = len(raw['Measurements'])
    FREQ = [raw['Measurements'][i]['fq'] for i in range(0, SIZE)] + FREQ
    RESISTANCE = [raw['Measurements'][i]['r'] for i in range(0, SIZE)] \
               + RESISTANCE
    REACTANCE = [raw['Measurements'][i]['x'] for i in range(0, SIZE)] \
               + REACTANCE

# Convert lists to Numpy arrays
FREQ = np.array(FREQ, dtype=float)
RESISTANCE = np.array(RESISTANCE, dtype=float)
REACTANCE = np.array(REACTANCE, dtype=float)

## Two steps to calculate the VSWR from the Resistance and Reactance:
# 1) Rho = sqrt( ((R - 50)^2 + X^2)/(R + 50)^2 + X^2))
RHO = np.sqrt(np.divide(np.square(RESISTANCE - 50) + np.square(REACTANCE),
                        np.square(RESISTANCE + 50) + np.square(REACTANCE)))

# 2) VSWR = (1 + Rho) / (1 - Rho)
VSWR = np.divide(1 + RHO, 1 - RHO)

# Sort the Frequencies and VSWR by Frequencies
VSWR = VSWR[np.argsort(FREQ)]
FREQ = FREQ[np.argsort(FREQ)]

# Create Horizontal lines at 1, 3, and 10
VSWR1 = len(FREQ)*[1]     # y = 1
VSWR3 = len(FREQ)*[3]     # y = 3
VSWR10 = len(FREQ)*[10]   # y = 10

#### Common Graphing Parameters
# Colours:   https://matplotlib.org/3.1.0/gallery/color/named_colors.html
CW = 'lightsalmon'
DIGI = 'lightskyblue'
PHONE = 'limegreen'
TV = 'plum'
BEACON = 'mistyrose'
MISC = 'gold'
OVERVIEW = 'black'
SWRCOLOUR = 'red'

# Vertical coordinates of labels
label1_y = -0.81    # Lower label
label2_y = -0.4     # Centre label
label3_y = 0.1      # Top label
label4_y = 8.1      # Very top band name

# Vertical Axis Marks
y_ticks = [1, 3, 5, 10]
```

```
####  FIRST PAGE:  2200m to 20m
# Axes: 7 graphs in a 8.5x11 page.
fig, ax = plt.subplots(nrows=8, figsize=(8.5, 11))

# Title on first [0] graph only
ax[0].set(title=TITLE)

# Axis labels
for i in range(0, 8):
    ax[i].set(ylabel='VSWR') # Set x-and y-axis labels
    # SWR Data
    ax[i].plot(FREQ, VSWR, color=SWRCOLOUR)   # Measured VSWR
    ax[i].plot(FREQ, VSWR1, ls='--', color='black', linewidth=0.5) # VSWR = 1
    ax[i].plot(FREQ, VSWR3, ls='--', color='black', linewidth=0.5) # VSWR = 3


### 2200m
i = 0                       # First graph
left = 0.1356               # Left edge
right = 0.1379               # Right edge
ax[i].set_xlim(left, right)  # Domain
ax[i].set_ylim(-1, 10)       # Range

# Frequency labels:
x1 = 0.1357
x2 = 0.1374
x3 = 0.1376
x4 = 0.1378
xlabels = [x1, x2, x3, x4]

# Bar Graphs
ax[i].broken_barh([(x1, 1.1*(x2-x1))], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, 1.1*(x3-x2))], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x3, x4-x3)], (-1, 2), facecolors=MISC)  # QRSS

# Labels
ax[i].text(left, label4_y, ' 2200m')
ax[i].text(0.13655, label2_y, 'CW', fontsize=7)
ax[i].text(0.13747, label2_y, 'Digi', fontsize=7)
ax[i].text(0.13765, label2_y, 'QRSS', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=20, ha='right')


### 630m
i = i + 1                       # First graph
```

```
left = 0.4716                # Left edge
right = 0.4794                # Right edge
ax[i].set_xlim(left, right)  # Domain
ax[i].set_ylim(-1, 10)        # Range

# Frequency labels:
x1 = 0.472
x2 = 0.475
x3 = 0.479
xlabels = [x1, x2, x3]

# Bar Graphs
ax[i].broken_barh([(x1, (x3-x1))], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, (x3-x2))], (-1, 1), facecolors=DIGI)

# Labels
ax[i].text(left, label4_y, ' 630m')
ax[i].text(0.4735, label2_y, 'CW', fontsize=7)
ax[i].text(0.4768, label2_y, 'Digi', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=20, ha='right')


### 160m
i = i + 1
left = 1.79
right = 2.01
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 1.800
x2 = 1.810
x3 = 1.840
x4 = 2.000
xlabels = [x1, x2, x3, x4]

# Bar Graphs
ax[i].broken_barh([(x1, 1.1*(x3-x1))], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x1, (x2-x1))], (-1, 1), facecolors=DIGI)
ax[i].broken_barh([(x3, (x4-x3))], (-1, 2), facecolors=PHONE)

# Labels
ax[i].text(left, label4_y, ' 160m')
ax[i].text(1.817, label2_y, 'CW', fontsize=7)
ax[i].text(1.915, label2_y, 'LSB', fontsize=7)
ax[i].text(1.801, label1_y, 'Digi', fontsize=7)
```

```
# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=25, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


#### 80m
i = i + 1
left = 3.475
right = 4.025
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 3.5
x2 = 3.58
x3 = 3.589
x4 = 3.6
x5 = 3.842
x6 = 3.845
x7 = 4
xlabels = [x1, x2, x4, x5, x7]

# Bar Graphs
ax[i].broken_barh([(x1, x3-x1)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, 0.003)], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x3, 1.1*(x4-x3))], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x4, x7-x4)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x5, x6-x5)], (-1, 2), facecolors=TV)

# Labels
ax[i].text(left, label4_y, ' 80m')
ax[i].text(3.536, label2_y, 'CW', fontsize=7)
ax[i].text(3.592, label2_y, 'D', fontsize=7)
ax[i].text(3.725, label2_y, 'LSB', fontsize=7)
ax[i].text(3.839, label2_y, 'TV', fontsize=7)
ax[i].text(3.91, label2_y, 'LSB', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=30, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


# 60m
```

```
i = i + 1
left = 5.327
right = 5.409
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 5.3305
x2 = 5.3465
x3 = 5.3515
x4 = 5.3715
x5 = 5.4035
xlabels = [x1, x1 + 0.0030, x2, x2 + 0.0030, x3, x3 + 0.0150,
           x4, x4 + 0.0030, x5, x5  + 0.0030]

# Bar Graphs
ax[i].broken_barh([(x1, 0.0030)], (0.33, 0.67), facecolors=CW)
ax[i].broken_barh([(x1, 0.0030)], (-0.33, 0.67), facecolors=PHONE)
ax[i].broken_barh([(x1, 0.0030)], (-1, 0.67), facecolors=DIGI)
ax[i].broken_barh([(x2, 0.0030)], (0.33, 0.67), facecolors=CW)
ax[i].broken_barh([(x2, 0.0030)], (-0.33, 0.67), facecolors=PHONE)
ax[i].broken_barh([(x2, 0.0030)], (-1, 0.67), facecolors=DIGI)
ax[i].broken_barh([(x3, 0.0150)], (0.33, 0.67), facecolors=CW)
ax[i].broken_barh([(x3, 0.0150)], (-0.33, 0.67), facecolors=PHONE)
ax[i].broken_barh([(x3, 0.0150)], (-1, 0.67), facecolors=DIGI)
ax[i].broken_barh([(x4, 0.0030)], (0.33, 0.67), facecolors=CW)
ax[i].broken_barh([(x4, 0.0030)], (-0.33, 0.67), facecolors=PHONE)
ax[i].broken_barh([(x4, 0.0030)], (-1, 0.67), facecolors=DIGI)
ax[i].broken_barh([(x5, 0.0030)], (0.33, 0.67), facecolors=CW)
ax[i].broken_barh([(x5, 0.0030)], (-0.33, 0.67), facecolors=PHONE)
ax[i].broken_barh([(x5, 0.0030)], (-1, 0.67), facecolors=DIGI)

# Labels
ax[i].text(left, label4_y, ' 60m')
ax[i].text(5.331, label3_y+0.19, 'CW', fontsize=5)
ax[i].text(5.331, label2_y+0.13, 'USB', fontsize=5)
ax[i].text(5.331, label1_y-0.1, 'Digi', fontsize=5)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=30, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.4f'))


# 40m
i = i + 1
left = 6.985
right = 7.315
ax[i].set_xlim(left, right)
```

```
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 7
x2 = 7.035
x3 = 7.04
x4 = 7.07
x5 = 7.125
x6 = 7.165
x7 = 7.175
x8 = 7.3
xlabels = [x1, x3, x4, x5, x6, x7, x8]

# Bar Graphs
ax[i].broken_barh([(x1, x3-x1)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, x3-x2)], (-1, 1), facecolors=DIGI)
ax[i].broken_barh([(x3, x8-x3)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x4, x5-x4)], (-1, 1), facecolors=DIGI)
ax[i].broken_barh([(x6, x7-x6)], (-1, 2), facecolors=TV)

# Labels
ax[i].text(left, label4_y, ' 40m')
ax[i].text(7.015, label2_y, 'CW', fontsize=7)
ax[i].text(7.0355, label1_y, 'D', fontsize=7)
ax[i].text(7.093, label1_y, 'Digi', fontsize=7)
ax[i].text(7.105, label3_y, 'LSB', fontsize=7)
ax[i].text(7.167, label2_y, 'TV', fontsize=7)
ax[i].text(7.23, label2_y, 'LSB', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=35, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


# 30m
i = i + 1
left = 10.097
right = 10.153
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 10.1
x2 = 10.13
x3 = 10.14
x4 = 10.15
xlabels = [x1, x2, x3, x4]
```

```
# Bar Graphs
ax[i].broken_barh([(x1, (x4-x1))], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, x3-x2)], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x3, x4-x3)], (-1, 1), facecolors=DIGI)

# Labels
ax[i].text(left, label4_y, ' 30m')
ax[i].text(10.115, label2_y, 'CW', fontsize=7)
ax[i].text(10.1345, label2_y, 'Digi', fontsize=7)
ax[i].text(10.1445, label3_y, 'CW', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=20, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.2f'))


# 20m
i = i + 1
left = 13.98
right = 14.37
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 14
x2 = 14.07
x3 = 14.073
x4 = 14.1005
x5 = 14.112
x6 = 14.230
x7 = 14.236
x8 = 14.350
xlabels = [x1, x2, x5, x6, x8]

# Bar Graphs
ax[i].broken_barh([(x1, x2-x1)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, x5-x2)], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x3, x4-x3)], (0, 1), facecolors=CW)
ax[i].broken_barh([(x5, x8-x5)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x6, x7-x6)], (-1, 2), facecolors=TV)

# Labels
ax[i].text(left, label4_y, ' 20m')
ax[i].text(14.033, label2_y, 'CW', fontsize=7)
ax[i].text(14.082, label3_y, 'CW', fontsize=7)
ax[i].text(14.087, label1_y, 'Digi', fontsize=7)
ax[i].text(14.17, label2_y, 'USB', fontsize=7)
ax[i].text(14.229, label2_y, 'TV', fontsize=7)
```

```
ax[i].text(14.28, label2_y, 'USB', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=20, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


### Print First Page
fig.tight_layout()
pdf_pages.savefig(fig)
#plt.savefig('CanBanPlan1.svg', transparent=False)



#### SECOND PAGE: 17m - 70cm
# Axes
fig2, ax = plt.subplots(nrows=7, figsize=(8.5, 11))

# Axis labels
for i in range(0, 7):
    ax[i].set(ylabel='VSWR') # Set x-and y-axis labels
    # SWR Data
    ax[i].plot(FREQ, VSWR, color=SWRCOLOUR)   # Measured VSWR
    ax[i].plot(FREQ, VSWR1, ls='--', color='black', linewidth=0.5) # VSWR = 1
    ax[i].plot(FREQ, VSWR3, ls='--', color='black', linewidth=0.5) # VSWR = 3


# 17m
i = 0
left = 18.062
right = 18.174
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 18.068
x2 = 18.095
x3 = 18.1
x4 = 18.11
x5 = 18.168
xlabels = [x1, x2, x3, x4, x5]

# Bar Graphs
ax[i].broken_barh([(x1, x3-x1)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, x3-x2)], (-1, 1), facecolors=DIGI)
ax[i].broken_barh([(x3, x4-x3)], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x4, x5-x4)], (-1, 2), facecolors=PHONE)
```

```
# Labels
ax[i].text(left, label4_y, ' 17m')
ax[i].text(18.082, label2_y, 'CW', fontsize=7)
ax[i].text(18.102, label2_y, 'Digi', fontsize=7)
ax[i].text(18.135, label2_y, 'USB', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=25, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


# 15m
i = i + 1
left = 20.975
right = 21.475
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 21
x2 = 21.07
x3 = 21.08
x4 = 21.083
x5 = 21.09
x6 = 21.125
x7 = 21.150
x8 = 21.340
x9 = 21.343
x10 = 21.450
xlabels = [x1, x2, x6, x7, x8, x10]

# Bar Graphs
ax[i].broken_barh([(x1, x7-x1)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, x6-x2)], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x2-.01, x3-x2+0.01)], (0, 1), facecolors=CW)
ax[i].broken_barh([(x4, x4-x3)], (0, 1), facecolors=CW)
ax[i].broken_barh([(x6, x7-x6)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x7, x10-x7)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x8, x9-x8)], (-1, 2), facecolors=TV)

# Labels
ax[i].text(left, label4_y, ' 15m')
ax[i].text(21.03, label2_y, 'CW', fontsize=7)
ax[i].text(21.095, label2_y, 'Digi', fontsize=7)
ax[i].text(21.13, label2_y, 'CW', fontsize=7)
ax[i].text(21.24, label2_y, 'USB', fontsize=7)
ax[i].text(21.337, label2_y, 'TV', fontsize=7)
ax[i].text(21.39, label2_y, 'USB', fontsize=7)
```

```
# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=20, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


# 12m
i = i + 1
left = 24.884
right = 24.996
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 24.89
x2 = 24.92
x3 = 24.925
x4 = 24.94
x5 = 24.975
x6 = 24.978
x7 = 24.990
xlabels = [x1, x2, x3, x4, x5, x7]

# Bar Graphs
ax[i].broken_barh([(x1, x2-x1)], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, x4-x2)], (-1, 2), facecolors=DIGI)
ax[i].broken_barh([(x3, x4-x3)], (0, 1), facecolors=CW)
ax[i].broken_barh([(x4, x7-x4)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x5, x6-x5)], (-1, 2), facecolors=TV)

# Labels
ax[i].text(left, label4_y, ' 12m')
ax[i].text(24.904, label2_y, 'CW', fontsize=7)
ax[i].text(24.9205, label2_y, 'Digi', fontsize=7)
ax[i].text(24.93, label3_y, 'CW', fontsize=7)
ax[i].text(24.956, label2_y, 'USB', fontsize=7)
ax[i].text(24.9755, label2_y, 'TV', fontsize=7)
ax[i].text(24.982, label2_y, 'USB', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=25, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))
```

```
# 10m
i = i + 1
left = 27.9
right = 29.8
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 28
x2 = 28.07
x3 = 28.1895
x4 = 28.2005
x5 = 28.3
x6 = 28.32
x7 = 28.68
x8 = 28.683
x9 = 29.3
x10 = 29.52
x11 = 29.7
xlabels = [x1, x2, x3, x6, x7, x9, x10, x11]

# Bar Graphs
ax[i].broken_barh([(x1, 1.1*(x6-x1))], (-1, 2), facecolors=CW)
ax[i].broken_barh([(x2, 1.1*(x6-x2))], (-1, 1), facecolors=DIGI)
ax[i].broken_barh([(x3, x5-x3)], (-1, 1), facecolors=BEACON)
ax[i].broken_barh([(x3, x4-x3)], (-1, 2), facecolors=BEACON)
ax[i].broken_barh([(x6, x11-x6)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x7, x8-x7)], (-1, 2), facecolors=TV)
ax[i].broken_barh([(x9, x10-x9)], (-1, 2), facecolors=MISC)

# Labels
ax[i].text(left, label4_y, ' 10m')
ax[i].text(28.015, label2_y, 'CW', fontsize=7)
ax[i].text(28.1, label1_y, 'Digi', fontsize=7)
ax[i].text(28.24, label3_y, 'CW', fontsize=7)
ax[i].text(28.2, label1_y, 'Beacon', fontsize=7)
ax[i].text(28.301, label1_y, 'D', fontsize=7)
ax[i].text(28.47, label2_y, 'USB', fontsize=7)
ax[i].text(28.665, label2_y, 'TV', fontsize=7)
ax[i].text(28.95, label2_y, 'USB', fontsize=7)
ax[i].text(29.39, label2_y, 'Sat', fontsize=7)
ax[i].text(29.59, label2_y, 'FM', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=25, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))
```

```
# 6m
i = i + 1
left = 49.78
right = 54.23
ax[i].set_xlim(left, right)
ax[i].set_ylim(-1, 10)

# Frequency labels:
x1 = 50
x2 = 50.1
x3 = 50.6
x4 = 51
x5 = 51.1
x6 = 52
x7 = 53
x8 = 54
xlabels = [x1, x3, x5, x6, x7, x8]

# Bar Graphs
ax[i].broken_barh([(x1, x8-x1)], (-1, 2), facecolors=PHONE)
ax[i].broken_barh([(x1, x2-x1)], (-1, 0.67), facecolors=CW)
ax[i].broken_barh([(x1, x2-x1)], (-0.33, 0.67), facecolors=BEACON)
ax[i].broken_barh([(x3, x4-x3)], (-1, 2), facecolors=MISC)
ax[i].broken_barh([(x4, x5-x4)], (-1, 1), facecolors=CW)
ax[i].broken_barh([(x5, x6-x5)], (-1, 1), facecolors=DIGI)
ax[i].broken_barh([(x5-0.003, 0.005)], (-1, 2), facecolors='black')
ax[i].broken_barh([(x6-0.003, 0.005)], (-1, 2), facecolors='black')
ax[i].broken_barh([(x7-0.003, 0.005)], (-1, 2), facecolors='black')

# Label
ax[i].text(left, label4_y, ' 6m')
ax[i].text(50.25, label2_y, 'USB', fontsize=7)
ax[i].text(50, label1_y-0.1, 'CW', fontsize=5)
ax[i].text(50, label2_y+0.1, 'Beac', fontsize=5)
ax[i].text(50.63, label2_y, 'Experimental', fontsize=6)
ax[i].text(51, label3_y, 'DX', fontsize=6)
ax[i].text(51, label1_y, 'CW', fontsize=6)
ax[i].text(51.35, label3_y, 'FM Simplex', fontsize=7)
ax[i].text(51.43, label1_y, 'Packet', fontsize=7)
ax[i].text(52.2, label2_y, 'FM Repeater Input', fontsize=7)
ax[i].text(53.2, label2_y, 'FM Repeater Output', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_ylim((-1, 10))
ax[i].set_yticks(y_ticks)
ax[i].set_xticklabels(xlabels, rotation=25, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.1f'))


# Re-define vertical coordinates of labels for 2m and 70cm
```

```
label1_y = 0.73    # Lower label
label2_y = 0.79    # Centre label
label3_y = 0.88      # Top label
label4_y = 1.85      # Very top band name

# 2m  https://wp.rac.ca/144-mhz-2m-page/
i = i + 1
left = 143.9
right = 148.1
ax[i].set_xlim(left, right)
ax[i].set_ylim(0.7, 2.1)

# Frequency labels:
ax[i].text(left, label4_y, ' 2m')
xlabels = [144, 148]

# Misc
ax[i].broken_barh([(144, 0.37)], (0.70, 0.3), facecolors=MISC)
ax[i].text(144.13, label2_y, 'Misc', fontsize=7)

# Digital Group 1
xlabels = xlabels + [144.370]
ax[i].broken_barh([(144.37, 0.12)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(144.37, label2_y, 'Digi', fontsize=7)

# Repeater Group 1 and 2
xlabels = xlabels + [144.51, 145.11]
ax[i].broken_barh([(144.51, 0.38)], (0.70, 0.3), facecolors=PHONE)
ax[i].broken_barh([(145.11, 0.38)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(145.115, label3_y, 'R$_1$', fontsize=7)
ax[i].text(145.115, label1_y, 'out', fontsize=7)
ax[i].text(144.515, label3_y, 'R$_1$', fontsize=7)
ax[i].text(144.515, label1_y, 'in', fontsize=7)
ax[i].text(145.27, label2_y, 'R$_2$ out', fontsize=7)
ax[i].text(144.67, label2_y, 'R$_2$ in', fontsize=7)
ax[i].broken_barh([(144.59, 0.02)], (0.70, 0.3), facecolors='white')
ax[i].broken_barh([(145.19, 0.02)], (0.70, 0.3), facecolors='white')

# Digital Repeater Group 1 and 2
xlabels = xlabels + [144.91, 145.51]
ax[i].broken_barh([(144.91, 0.18)], (0.70, 0.3), facecolors=DIGI)
ax[i].broken_barh([(145.51, 0.18)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(144.91, label3_y, 'R$_2$', fontsize=7)
ax[i].text(144.91, label1_y, 'in', fontsize=7)
ax[i].text(145.51, label3_y, 'R$_2$', fontsize=7)
ax[i].text(145.51, label1_y, 'out', fontsize=7)
ax[i].text(145.01, label3_y, 'R$_1$', fontsize=7)
ax[i].text(145.01, label1_y, 'out', fontsize=7)
ax[i].text(145.61, label3_y, 'R$_1$', fontsize=7)
ax[i].text(145.61, label1_y, 'in', fontsize=7)

# Repeater Group 3
```

```
xlabels = xlabels + [146.02, 146.62]
ax[i].broken_barh([(146.02, 0.36)], (0.70, 0.3), facecolors=PHONE)
ax[i].broken_barh([(146.62, 0.36)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(146.1, label2_y, 'R$_3$ in', fontsize=7)
ax[i].text(146.7, label2_y, 'R$_3$ out', fontsize=7)

# Repeater Group 4
xlabels = xlabels + [147, 147.6]
ax[i].broken_barh([(147, 0.38)], (0.70, 0.3), facecolors=PHONE)
ax[i].broken_barh([(147.6, 0.38)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(147.1, label2_y, 'R$_4$ out', fontsize=7)
ax[i].text(147.7, label2_y, 'R$_4$ in', fontsize=7)

# Digital Simplex
xlabels = xlabels + [145.71]
ax[i].broken_barh([(145.71, 0.08)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(145.71, label2_y, 'Sx', fontsize=7)

# Satellite
xlabels = xlabels + [145.8]
ax[i].broken_barh([(145.8, 0.2)], (0.70, 0.3), facecolors=MISC)
ax[i].text(145.85, label2_y, 'Sat', fontsize=7)

# FM Simplex
xlabels = xlabels + [146.415]
ax[i].broken_barh([(146.415, 0.18)], (0.70, 0.3), facecolors=PHONE)
ax[i].broken_barh([(147.6, 0.38)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(146.415, label2_y, 'Sx', fontsize=7)
ax[i].text(147.7, label2_y, 'R$_4$ in', fontsize=7)

# Internet Linked Simplex
xlabels = xlabels + [147.42]
ax[i].broken_barh([(147.42, 0.165)], (0.70, 0.3), facecolors=MISC)
ax[i].text(147.42, label3_y, 'Net Lk', fontsize=7)
ax[i].text(147.42, label1_y, 'Sx', fontsize=7)

# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_xticklabels(xlabels, rotation=40, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))


# 70cm  https://wp.rac.ca/144-mhz-2m-page/
i = i + 1
left = 429.5
right = 450.5
ax[i].set_xlim(left, right)
ax[i].set_ylim(0.7, 2.1)

# Frequency labels:
ax[i].text(left, label4_y, ' 70cm')
```

```
xlabels = [430, 450]

# Packet Trunked Repeaters
xlabels = xlabels + [439.05]
ax[i].broken_barh([(430.05, 0.9)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(430.05, label3_y, 'Packet', fontsize=7)
ax[i].text(430.05, label1_y, 'Output', fontsize=7)
ax[i].broken_barh([(439.05, 0.9)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(439.05, label3_y, 'Packet', fontsize=7)
ax[i].text(439.05, label1_y, 'Input', fontsize=7)

# Not allocated
xlabels = xlabels + [431]
ax[i].broken_barh([(431, 0.475)], (0.70, 0.3), facecolors='grey')

# Misc
xlabels = xlabels + [431.5]
ax[i].broken_barh([(431.5, 1.5)], (0.70, 0.3), facecolors=MISC)
ax[i].text(432, label2_y, 'Misc', fontsize=7)

# Digi Output
xlabels = xlabels + [433.025]
ax[i].broken_barh([(433.025, 0.975)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(433.025, label3_y, 'R$_1$', fontsize=7)
ax[i].text(433.025, label1_y, 'Output', fontsize=7)

# Digi Input
xlabels = xlabels + [438.025]
ax[i].broken_barh([(438.025, 0.975)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(438.025, label3_y, 'R$_1$', fontsize=7)
ax[i].text(438.025, label1_y, 'Input', fontsize=7)

# Repeater Output
xlabels = xlabels + [434.025]
ax[i].broken_barh([(434.025, 0.95)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(434.025, label3_y, 'R$_1$', fontsize=7)
ax[i].text(434.025, label1_y, 'Output', fontsize=7)

# Repeater Input FIXME
#xlabels = xlabels + [439.025]
ax[i].broken_barh([(439.025, 0.95)], (0.70, 0.15), facecolors=PHONE)
#ax[i].text(439.025, label3_y, 'R$_1$', fontsize=7)
#ax[i].text(439.025, label1_y, 'Input', fontsize=7)

# Sat
xlabels = xlabels + [435]
ax[i].broken_barh([(435, 3)], (0.70, 0.3), facecolors=MISC)
ax[i].text(436.3, label2_y, 'Sat', fontsize=7)

# Digital and link repeaters, except where used by IC Output
xlabels = xlabels + [440.025]
ax[i].broken_barh([(440.025, 0.925)], (0.70, 0.3), facecolors=DIGI)
```

```
ax[i].text(440.025, label3_y, 'Digi', fontsize=7)
ax[i].text(440.025, label1_y, 'Output', fontsize=7)

# Digital and link repeaters, except where used by IC Input
xlabels = xlabels + [445.025]
ax[i].broken_barh([(445.025, 0.925)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(445.025, label3_y, 'Digi', fontsize=7)
ax[i].text(445.025, label1_y, 'Input', fontsize=7)

# Simplex Point-to-Point links
xlabels = xlabels + [441]
ax[i].broken_barh([(441, 0.975)], (0.70, 0.3), facecolors=DIGI)
ax[i].text(440.9, label3_y, 'Simplex', fontsize=7)
ax[i].text(441, label1_y, 'Link', fontsize=7)

# FM Repeaters Output
xlabels = xlabels + [442]
ax[i].broken_barh([(442, 0.975)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(442, label3_y, 'R$_2$', fontsize=7)
ax[i].text(442, label1_y, 'Output', fontsize=7)

# FM Repeaters Input
xlabels = xlabels + [447]
ax[i].broken_barh([(447, 0.975)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(447, label3_y, 'R$_2$', fontsize=7)
ax[i].text(447, label1_y, 'Input', fontsize=7)

# FM Repeaters Output
xlabels = xlabels + [443.025]
ax[i].broken_barh([(443.025, 1.95)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(443.025, label3_y, 'R$_3$', fontsize=7)
ax[i].text(443.025, label1_y, 'Output', fontsize=7)

# FM Repeaters Input
xlabels = xlabels + [448.025]
ax[i].broken_barh([(448.025, 1.95)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(448.025, label3_y, 'R$_3$', fontsize=7)
ax[i].text(448.025, label1_y, 'Input', fontsize=7)

# FM Simplex
xlabels = xlabels + [446]
ax[i].broken_barh([(446, 0.975)], (0.70, 0.3), facecolors=PHONE)
ax[i].text(446, label2_y, 'Simplex', fontsize=7)


# Axis
ax[i].set_xlim((left, right))
ax[i].set_xticks(xlabels)
ax[i].set_xticklabels(xlabels, rotation=40, ha='right')
ax[i].xaxis.set_major_formatter(mtick.FormatStrFormatter('%1.3f'))
```

```
# Print second page
fig2.tight_layout()
pdf_pages.savefig(fig2)
#plt.savefig('CanBanPlan2.svg', transparent=False)
#plt.savefig('CanBanPlan2.pdf', transparent=False)


## Third Page

# Axis: 2 graphs on a 11x8.5 sheet (landscape)
fig3, ax = plt.subplots(nrows=2, figsize=(11, 8.5))


# HF
i = 0
ax[i].set(title='HF')
left = 0
right = 55
y_ticks = [1, 3, 5, 10, 15, 20, 25]

# Axis
ax[i].set_xlim((left, right))
ax[i].set_ylim(-0.3, 25)
ax[i].set_yticks(y_ticks)

# 160m
ax[i].broken_barh([(1.8, 0.2)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(1.0, -1, '160m', fontsize=7)

# 80m
ax[i].broken_barh([(3.5, 0.5)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(3.2, -1, '80m', fontsize=7)

# 60m
ax[i].broken_barh([(5.3305, 0.076)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(5.05, -1, '60m', fontsize=7)

# 40m
ax[i].broken_barh([(7, 0.3)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(6.8, -1, '40m', fontsize=7)

# 30m
ax[i].broken_barh([(10.1, 0.05)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(10.15, label2_y-0.7, '← 30m', fontsize=7)

# 20m
ax[i].broken_barh([(14, 0.3)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(13.7, -1, '20m', fontsize=7)

# 17m
ax[i].broken_barh([(18.068, 0.1)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(17.8, -1, '17m', fontsize=7)
```

```
# 15m
ax[i].broken_barh([(21, 0.4)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(20.7, -1, '15m', fontsize=7)

# 12m
ax[i].broken_barh([(24.89, 0.1)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(24.6, -1, '12m', fontsize=7)

# 10m
ax[i].broken_barh([(28, 1.7)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(28, -1, '10m', fontsize=7)

# 50m
ax[i].broken_barh([(50, 4)], (-0.3, 1.3), facecolors=OVERVIEW)
ax[i].text(51.5, -1, '6m', fontsize=7)


# SWR Data
ax[i].plot(FREQ, VSWR, color=SWRCOLOUR)   # Measured VSWR
ax[i].plot(FREQ, VSWR1, ls='--', color='black', linewidth=0.5) # VSWR = 1
ax[i].plot(FREQ, VSWR3, ls='--', color='black', linewidth=0.5) # VSWR = 3
ax[i].plot(FREQ, VSWR10, ls='--', color='black', linewidth=0.5) # VSWR = 10
ax[i].set(ylabel='VSWR', xlabel='Freq [MHz]') # Set x-and y-axis labels


# VHF / UHF
i = i + 1
ax[i].set(title='VHF and UHF')
left = 100
right = 500
y_ticks = [1, 1.5, 2, 3, 5]
ax[i].set_xlim(left, right)
ax[i].set_ylim(0.8, 5)
ax[i].set_yticks(y_ticks)

# 2m
ax[i].broken_barh([(144, 4)], (0.80, 0.2), facecolors=OVERVIEW)
ax[i].text(138, 0.7, '2m', fontsize=7)

# 135cm
ax[i].broken_barh([(222, 3)], (0.80, 0.2), facecolors=OVERVIEW)
ax[i].text(220, 0.7, '135cm', fontsize=7)

# 70cm
ax[i].broken_barh([(430, 20)], (0.80, 0.2), facecolors=OVERVIEW)
ax[i].text(430, 0.7, '70cm', fontsize=7)

# SWR Data
ax[i].plot(FREQ, VSWR, color=SWRCOLOUR)   # Measured VSWR
ax[i].plot(FREQ, VSWR1, ls='--', color='black', linewidth=0.5) # VSWR = 1
ax[i].plot(FREQ, VSWR3, ls='--', color='black', linewidth=0.5) # VSWR = 3
```

```
ax[i].set(ylabel='VSWR', xlabel='Freq [MHz]') # Set x-and y-axis labels


# Print Third page
fig3.tight_layout()
pdf_pages.savefig(fig3)
pdf_pages.close()
```