Contents

Front Cover 01-1
Title Page 01-2
System Photos 02-1
System Description 04-1
System Block Diagram 05-1
Radio Modification, Rpi Controller, SVXlink Modules 06-1
Audio issues and Resolutions 08-1
Internet (Network) Routing 10-1
Software Errata 11-1
Svxlink Configuration Files 12-1
Icom Radio Description, Schematics and Mods 17-1
Interface Board Layout 26-1
Icom OPC-617 Interface Cable (Modified) Schematic 27-1
Interface Board Switching Schematic 28-1
Interface Board Audio Schematic 29-1
Radio Front Panel Mic Connections 30-1
Icom Microphone (Not Used) Schematic 31-1
Rpi Power Down Push Button Schematic 32-1
Rpi GPIO Connector Layout 33-1
System Start-up/Shut-Down Procedure, System D, “C” Code 35-2
Updating the Rpi 36-1
Sound Card Configuration 36-1
How to Clone the Rpi SD Card 37-1
How to Back-Up the Rpi SD Card 38-1
SD Card Preservation 39-1
Setting up the GPIO Ports 41-1
Sound Configuration 43-1
Installing the Raspbian Linux Operating System 44-1
Installing SVXlink 45-1
Custom SVXlink Configuration 48-1
SVXlink Hassles and Issues (Good additional info) 51-1



York Radio Club Echolink Node

Bob Lund: WB9HYB
May - 2018

W9YRC
York Radio Club
Echolink Node

2018

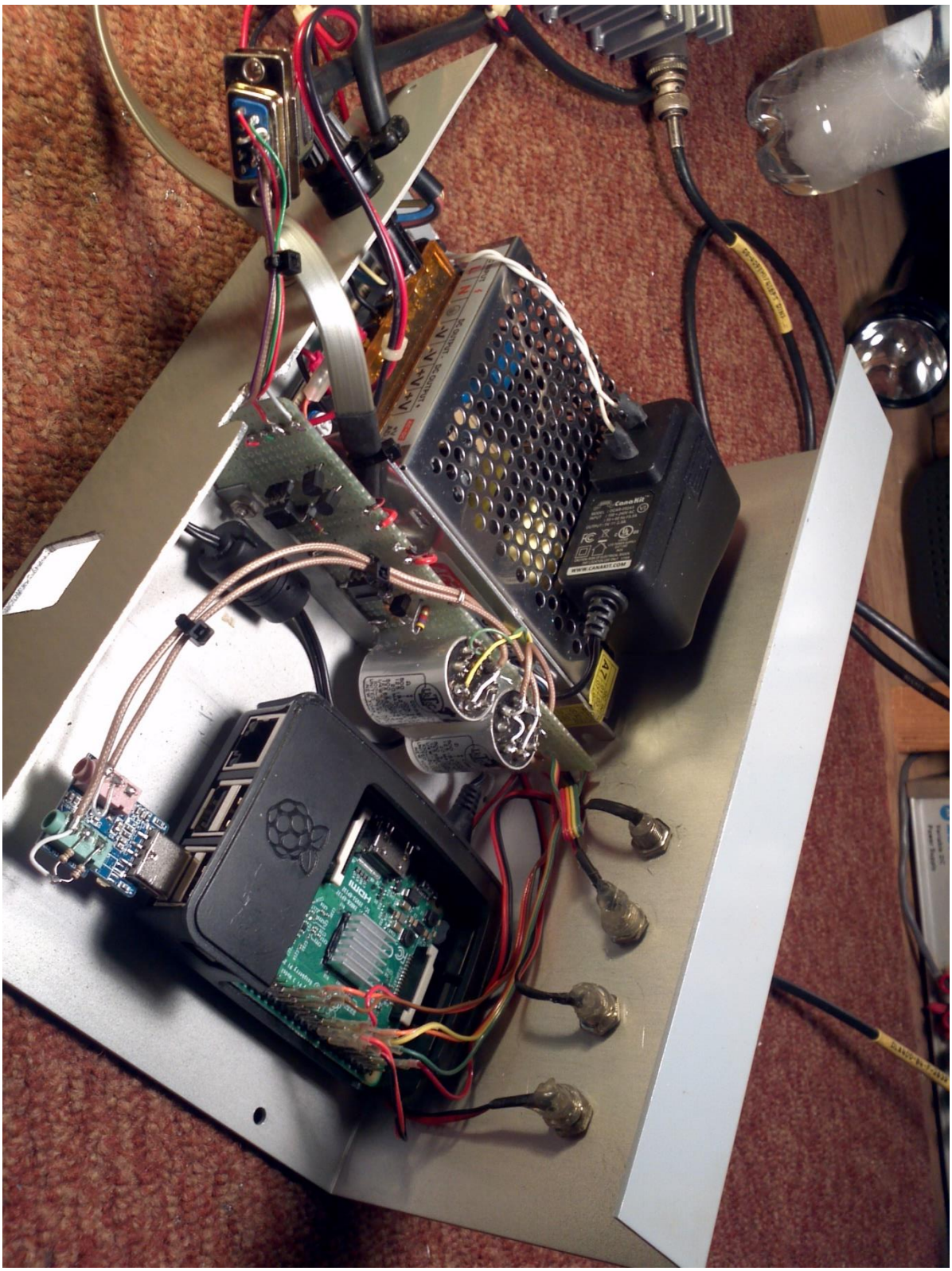
Bob Lund – WB9HYB

File Name: Echolink Node [W9YRC].docx

- [First Draft 3/16/18]-
- [Added audio details: 4/10/18]
- [Added network description: 4/28/18]







Icom F420 UHF Mobile And Echolink

[March 2018]

This document set describes the process for packaging an Icom model F420 UHF mobile radio with a Raspberry Pi single board computer to facilitate an Echolink node.

The implementation described in this document consists of the following hardware and software components:

- "Open Repeater" control software front end.
- "SVX Link" main control software.
- Raspberry Pi model 3 with Debian Linux.
- "Plugable" Sound Card.
- Fabricated Interface board.

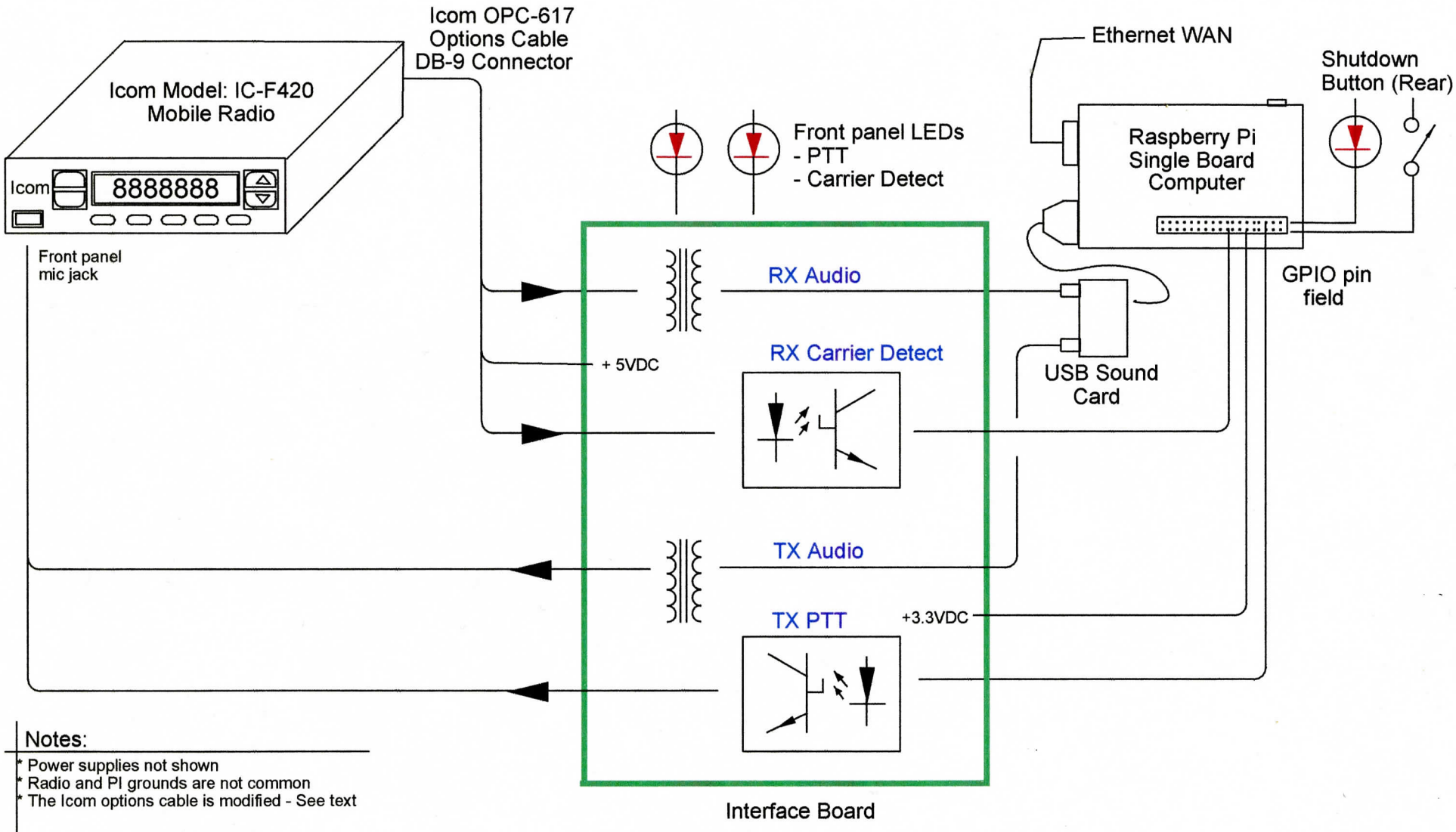
Icom F420 Radio --

To operate as an Echolink node, the radio must be interfaced to an external computer and modified to operate in a remote-controlled environment. The controller or single board computer applies the push-to-talk transmit/receive switching and via a USB sound card, provides the transmit and receive audio.

With only two exceptions, all the remote-control functions associated with the radio are available through the addition of an Icom model #: OPC-617 option cable which extends the internal functions of the radio to the rear of the chassis via a short umbilical cord terminated in a DB-9 connector. The two exceptions which require modifications to the options cable are the requirements for a "Carrier Operated Squelch" (COS) point and the provision for a current limited +5VDC for switching functions. Push-to-Talk (PTT) and transmit audio are applied through the front panel microphone connector

Interface Board –

An interface board was fabricated to facilitate full isolation of the Radio and the Raspberry Pi controller. This is typically required to eliminate any potential ground loops and/or audio level issues. Also, the logic switching levels between the radio and the PI are substantially different and are isolated with optical couplers. The interface board consists of optical isolators for the COS and transmit logic and coupling transformers for the TX and RX audio lines.



(1-50)

Notes:

* Power supplies not shown
* Radio and PI grounds are not common
* The Icom options cable is modified - See text

Scale:	8.5 X 11 [A]	York Radio Club Echolink Node	
Eng:	B. Lund WB9HYB	System Block Diagram	
Date:	March 2018	File: System Block Diagram	Page:

Radio Modifications –

As mentioned above, 2-lines in the OPC-617 option cable are otherwise unused and were sacrificed to extend the COS and +5VDC out of the radio to the external interface switching board:

- A +5VDC point was located on the main board in the radio. A series 100-Ohm resistor is inserted and this DC point is extended to pin #1 on the DB-9 connector. The function of the 100-Ohm resistor is to add a level of current limiting protection to the radio should the line be accidentally shorted or subjected to any external interface failure.
- After close examination of the radio circuitry it was discovered that a "Carrier Operated Squelch" (COS) logic level existed within the radio's internal audio path. This level was discovered at pin#13 on IC-11. A 10K-Ohm resistor was added to this point and extended out to the DB-9 connector. This 10K resistor serves the same protective theory as described for the 5V line above. In retrospect, this is a very desirable logic function. I'm surprised ICOM didn't provide this on the OPC-617 option cable. The newly discovered COS line provides a logic "high" when *both* the squelch opens *and* a valid PL tone is received. In this mode, the microphone hook switch needs to hang in the mic clip to enable the PL receive function. The PL enable line is grounded and enabled through the front panel connector cable so the mic hang-up is not required.

Raspberry PI Controller –

This is the main controller for the Echolink node and is powered by the standard Debian Linux operating system. The "SVXlink" software package provides the core logic for the system. SVXlink is quite an extensive software package and provides a large general-purpose library of functions for radio control. Most of these functions are not used in this application. To simplify the Echolink functionality, a software package called "Open Repeater" is layered on top of SVXlink. Open Repeater sets up the configuration file(s) required for SvxLink. Although Openrepeater is installed on this system, it is not used. The full description of these software packages is beyond the scope of this document. For further information, the reader may wish to visit these web sites:

https://openrepeater.com/

https://github.com/sm0svx/svxlink/wiki

Also, there is a wealth of information on the web regarding these software packages and installation details.

SVXlink Modules --

What we need to do is explore and change a few of the SVXlink options. The only configuration files that need to be edited are:

/etc/openrepeater/svxlink/svxlink.conf
/etc/openrepeater/svxlink/svxlink.d/ModuleEchoLink.conf

Notice that there is a reference to "openrepeater" in these file locations. Under normal circumstances, openrepeater is not used. Openrepeater is an application that rides on top of SVXlink and was supposedly designed as a simplified user interface for SVXlink. In most cases, *openrepeater just got in the way!* Don't use it! In fact, if you try Openrepeater and change anything it will completely overwrite your previous config adjustments. Although it is installed in this iteration it is *not used*.

configure /etc/svxlink/svxlink.conf

```
[GLOBAL] Section
LOGICS=SimplexLogic

[SimplexLogic] Section
CALLSIGN

[Rx1] Section
AUDIO_DEV=alsa:plughw:1

[Tx1] Section
AUDIO_DEV=alsa:plughw:1
PTT_TYPE=GPIO
PTT_PORT=gpio17
```

Turn off the "Long" and "Short" ID (Set to "0" or comment out) – Not needed. Keeps SVXlink from sending id's. The main repeater takes care of this.

GPIO Squelch input pin
In this build, I used a negative going logic level on this pin. I didn't realize until later that the default input for SVXlink wants a high going logic level on the GPIO pin for open squelch. Not a problem – Buried in the docs for the "svxlink.conf" file there is a note that says you can add a "!" character ahead of the pin designator to reverse the logic.to a low going level. See the svxlink.conf file listing.

As far as GPIO pins, look in: sys/class/gpio for detailed gpio files and info – Neat!!

audio coming into the Pi from the radio, but not going out the internet - No idea what's up with this issue yet. doing a tcpdump shows the software isn't even trying to send. Everything else works (inet → repeater, inet → inet).

FIXED - *Use SimplexLogic instead of RepeaterLogic*. The basic issue is RepeaterLogic assumes a hard-linked repeater connection rather than a remote-link connection as described on the Echolink site. As a result, svxlink wants to retransmit back out to the repeater, rather than send the received audio out the internet.

Audio issues and Resolutions --

Researching several web pages regarding audio issues that other users have experienced shows that just about every installation is unique. Although every implementation is unique, there are a few common areas which require detailed attention.

USB Sound Card - Hardware

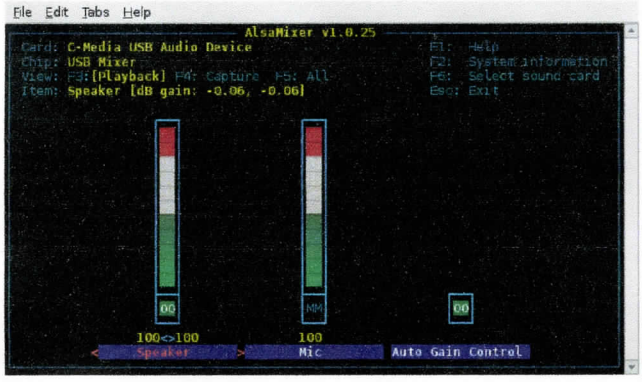
In this installation a "Plugable" style sound card is used as the audio interface between the radio and the Rpi computer. This small USB card plugs directly into one of the Rpi's USB ports (It doesn't matter which one). This card doesn't have any hardware adjustments but here are a few details regarding the audio I/O connections:

- I removed the USB card from its housing. This was done so direct connections could be made to the audio points on the card without the need for external plugs. I wasted a good deal of time struggling with intermittent audio while trying to use those cheapie connectors.
- Contrary to how it appears, the microphone input (Pink) is a mono connection. As shown on the diagram only pins #1 (Gnd.) and #2 (Audio hot) are used. Pin #3 has no connection. This is how this sound card is designed and may be different on other card models.
- The output headphone connector (Green) is a stereo connection. Pin #1 is the common (Gnd.) and pins #2 and #3 are the left and right stereo outputs.
- Refer to the interface board [Audio] drawing for further details on these connections.

USB Sound Card – Software

Once the hardware connections are completed, we need to look at how the Rpi sees the audio. This is done through a Linux application called "Alsamixer." This application provides various gain settings for the audio channels. There is a multitude of internet documents detailing this application but here are a few additional details specific to this project:

- Run Alsamixer from the Linux SSH command line. You should see a screen like this:



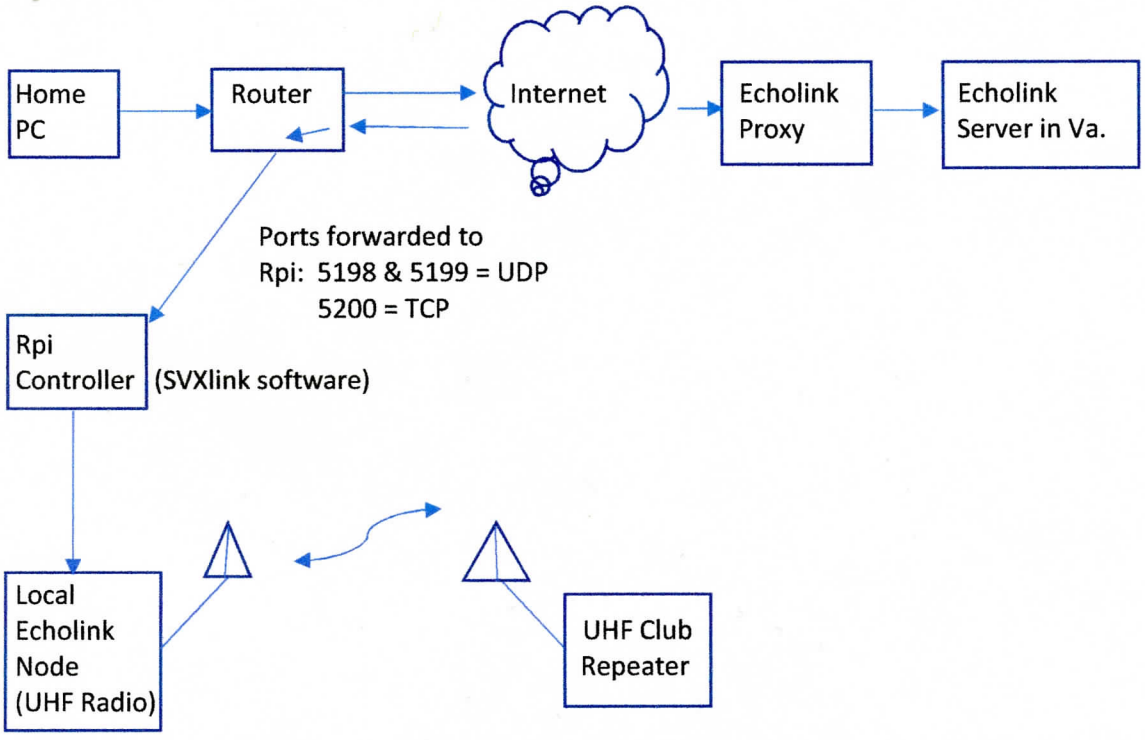
Enter <F5> to expand the display to show the speaker and mic vertical level slider bars. Make sure all the slider bar items are not muted – i.e., the little box area under each slider bar shows either an "MM" for muted or "OO" for open. Use the arrow keys to select each

slider bar and then use the keyboard "M" key to toggle between mute and open. All the slider bars should show "OO" for open.

- Once your Alsamixer options seems adequate, exit Alsamixer and run "Alsactl store" to save your settings.
- If SVXlink is running, audio is echoed from the USB soundcard mic input to the headphone output. This is handy for initial testing. With a set of headphones plugged into the green output connector you can hear the receiver audio that is applied to the soundcard mic input. Since the mic input is mono, this audio comes out on both the left and right stereo channels simultaneously.
- When you connect to Echolink, the nice lady's voice (Heather) that announces the connection only comes out of one of the stereo channels. So, you need to select the correct stereo channel that has both the RX audio and the voice for application to the transmitter. If you use the wrong channel, you'll get the RX audio but not the lady's voice announcements.

- **Network:**

To have the system run on a home network you need to "port forward" 3-connections in your router: UDP -5189, 5199 and TCP – 5200. Once this is done, you'll no longer have use of the Echolink application on your PC because now, those 3-ports in the router point to the Rpi controller and not to your home networked PC. The resolution here is to connect through one of Echolink's proxies. There is a drop-down menu selection for this in the Echolink program. These are Ham controlled proxies and are sometimes busy or a little sketchy in operation; Just keep trying different ones until a good connection is established.



You'll note in the above sketch that SVXlink (running on the Rpi controller) is not configured as a repeater controller. It is running in the "Simplex" mode. In more conventional terms, this system is nothing more than a remote base station connected over the internet. Of course, any Ham can connect to this base station node with their home PC or a Cell phone through the internet via the Echolink server in VA.

As a final note, always execute a controlled shutdown: "sudo shutdown -h now" to prevent corrupting the memory card!

Bob – WB9HYB
[4/28/18]

Software errata –

Some of the following points have already been covered. These are just some random notes that I jotted down while developing and working on this project and are in no particular order:

- **These are the three (3) main configuration files:**

  /etc/openrepeater/svxlink/svxlink.conf
  /etc/openrepeater/svxlink/svxlink_gpio.conf
  /etc/openrepeater/svxlink/svxlink.d/ModuleEchoLink.conf

  After editing, simply: "sudo systemctl restart svxlink" instead of a full reboot (sudo reboot).

- **Log file:**

  /var/log/svxlink

  To watch the log file in real time while the system is running:

  tail -f /var/log/svxlink

- **[Audio] Alsa mixer:**

  After alsamixer settings: "alsactl store " to save (lock) the settings
  "aplay -l" or "arecord -l" to list devices

  While running, svxlink grabs the USB sound device(s). You need to stop svxlink to release the sound card for testing:

  Sudo systemctl disable svxlink -then-
  Sudo systemctl restart svxlink

  You can also "systemctl stop, start, show, status svxlink" or other services

  To test the usb sound card, disable svxlink as above then use:
  "speaker-test -D hw:0 -c2 -twave" The nice lady will speak "Front-left, Front-right" over and over.
  Just "Speaker-test" will output white noise only.

  Finally, execute: "alsactl store" to save settings

- **Misc. commands:**
  - sudo find / -iname *.* = searches for a file(s)
  - top = Shows running services. SVXlink is at the top.

Svxlink.conf

[May 2, 2018]

pi@rpi-repeater:/etc/openrepeater/svxlink/svxlink.conf

```
# ****************************************************************************
# *
# *                    Svxlink Main System Configuration File
# *                    ======================================
# * Revision History:
# * ----------------
# * 04/16/18 - WB9HYB - Converted the logistics from "Repeater" to "Simplex"
# *              and removed extraneous repeater config parms.
# * 04/18/18 - WB9HYB - Commented out a few of the RX squelch lines.
# * 04/24/18 - WB9HYB - Removed all references/lines to "DTMF" to simplify the
# *              file.
# ****************************************************************************

[GLOBAL]
MODULE_PATH=/usr/lib/arm-linux-gnueabihf/svxlink
LOGICS=SimplexLogic
CFG_DIR=svxlink.d
TIMESTAMP_FORMAT="%c"
CARD_SAMPLE_RATE=16000

[SimplexLogic]
TYPE=Simplex
RX=Rx1
TX=Tx1
MODULES=ModuleEchoLink
CALLSIGN=W9YRC-R
# Long and short id's set to "Zero" to disable them. Otherwise it transmits
# to the repeater = irritating and not needed. WB9HYB, 4/7/18
SHORT_IDENT_INTERVAL=0
LONG_IDENT_INTERVAL=0
EVENT_HANDLER=/usr/share/svxlink/events.tcl
DEFAULT_LANG=en_US
MACROS=Macros
FX_GAIN_NORMAL=-3
FX_GAIN_LOW=-12
```

```
# Port 1 Receive
#********************
[Rx1]
TYPE=Local
AUDIO_DEV=alsa:plughw:0
AUDIO_CHANNEL=1
SQL_DET=GPIO
# Added a "!" ahead of the following gpio pin number
# to invert the logic. Otherwise, this pin will require
# a high level on open squelch.  WB9HYB, 4/7/18
#GPIO_SQL_PIN=gpio17
GPIO_SQL_PIN=!gpio17
#SQL_HANGTIME=10
#SQL_START_DELAY=1
#SQL_DELAY=10
#DEEMPHASIS=1
#PEAK_METER=0

# Port 1 Transmit
#*******************
[Tx1]
TYPE=Local
AUDIO_DEV=alsa:plughw:0
AUDIO_CHANNEL=1
PTT_TYPE=GPIO
PTT_PORT=GPIO
PTT_PIN=gpio27
PTT_HANGTIME=1000
TIMEOUT=300
TX_DELAY=750
```

ModuleEchoLink.conf

[May 2, 2018]

pi@rpi-repeater:/etc/openrepeater/svxlink/svxlink.d/ModuleEchoLink.conf

```
[ModuleEchoLink]
NAME=EchoLink
ID=2
TIMEOUT=60
SERVERS=servers.echolink.org
CALLSIGN=W9YRC-R
PASSWORD=xxxx
SYSOPNAME=York Radio Club
LOCATION=Elmhurst, IL.
MAX_QSOS=1
MAX_CONNECTIONS=1
LINK_IDLE_TIMEOUT=300
DESCRIPTION="Welcome to the York Radio Club EchoLink connection."
USE_GSM_ONLY=1
```

svxlink_gpio.conf

[April 18, 2018]

File location:
/etc/openrepeater/svxlink/svxlink_gpio.conf

```
##############################################################################
#
#       Configuration file for the SvxLink server GPIO Pins
#
##############################################################################

#Set what gpio pins point out
GPIO_OUT_PIN="27"
#Set what gpio pins point in
GPIO_IN_PIN="17"
#Set what gpio pins need to be set low
#GPIO_LOW_PIN="17" #LOW STATE IS CURRENTLY HARD CODE FOR INPUT/COS PINS
```

System Schematics And information

VHF MOBILE TRANSCEIVERS
IC-F310 IC-F320

UHF MOBILE TRANSCEIVERS
IC-F410 IC-F420

SIMPLE, POWERFUL COMMUNICATIONS

Meets MIL-STD 810

ETS : IC-F310/F410
FCC : IC-F320/F420



These mobile radios provide the operating simplicity to keep your eyes on the road—not the front panel display! But don't let the lack of buttons and switches fool you, there are enough features and functions to meet most commercial demands.

Trunking capabilities

Compatible with the SmarTrunk II™, or the LTR®* trunking systems, when the optional UT-105 for SmarTrunk II™, or UT-111 for LTR® trunking is installed.
*IC-F420 only

Flexible selective calling

CTCSS/DTCS encoder/decoder with user programmable CTCSS, DTCS capabilities, and DTMF, 2/5-tone encoder are built-in. The optional UT-96 (for 2/5-tone) allows decode functions. In addition, special tone code, for group, and individual calls, and more features are available to support flexible, perfect selective calling operations.

Simplified operation

Easy-to-operate—large, clearly labelled switches provide up/down adjustment of channels and volume. The remaining 4 front panel switches (not including [PWR]) are dealer-programmable so that operation can be customized and simplified to meet specific needs. In addition, key backlight helps during night time operation.

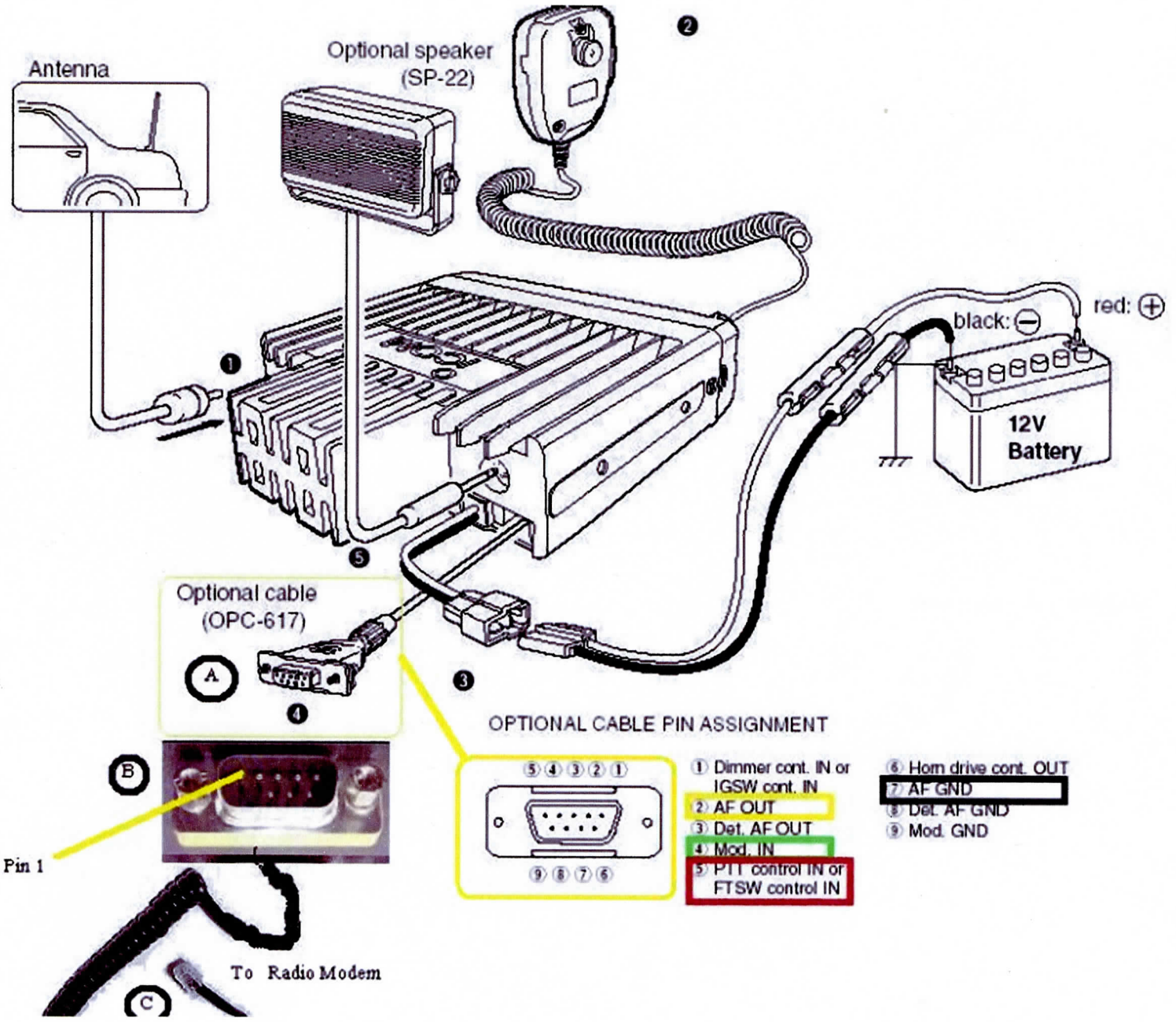
Operating security

Optional voice scrambler units, UT-109 and UT-110, for secure communications, are available.

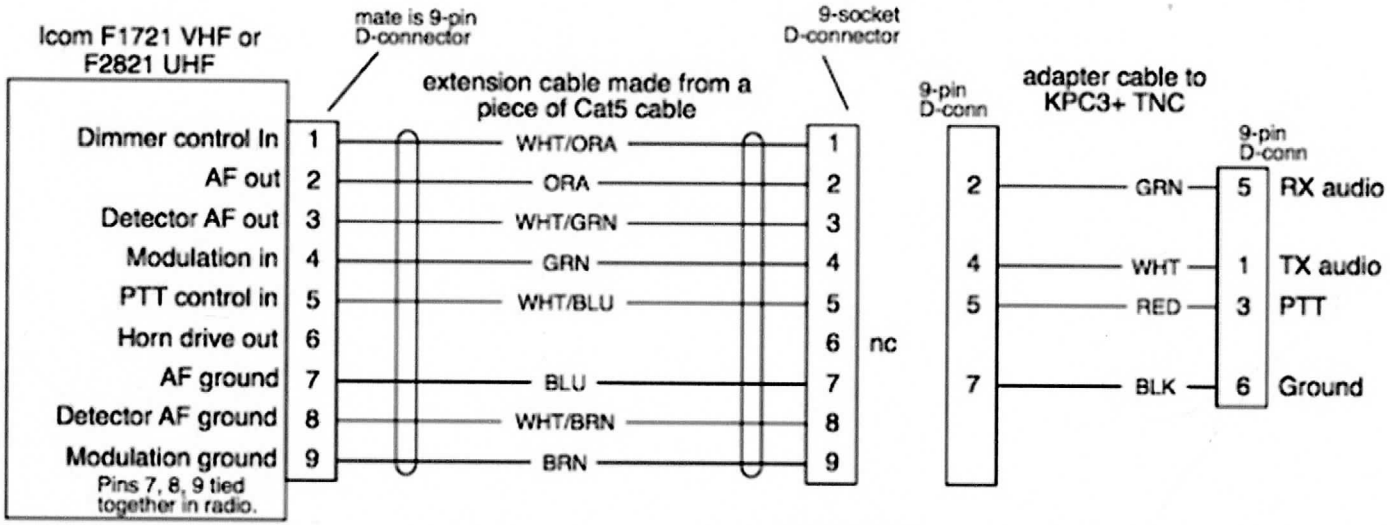
Other features

- Up to 32 memories with 5-character memory name indication
- 5 + 2 digit alphanumeric display
- Selectable channel spacing (depending on version)
- 51 frequencies for CTCSS (plus 1 programmable)
- Selectable 5-tone format
- 7 autodialing memories for DTMF and more...

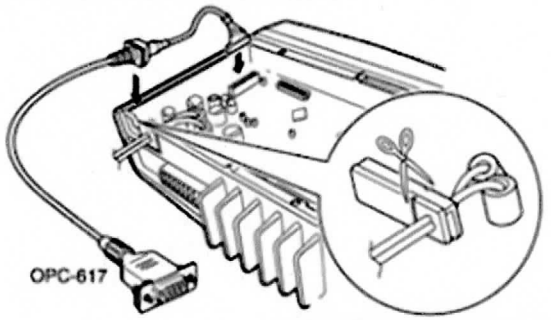



Antenna
Optional speaker
(SP-22)
❷
❶
❺
black: ⊖
red: ⊕
12V
Battery
Optional cable
(OPC-617)
Ⓐ
❹
❸
OPTIONAL CABLE PIN ASSIGNMENT
⑤④③②①
⑨⑧⑦⑥
① Dimmer cont. IN or IGSW cont. IN
② AF OUT
③ Det. AF OUT
④ Mod. IN
⑤ PTT control IN or FTSW control IN
⑥ Horn drive cont. OUT
⑦ AF GND
⑧ Det. AF GND
⑨ Mod. GND
Ⓑ
Pin 1
To Radio Modem
Ⓒ

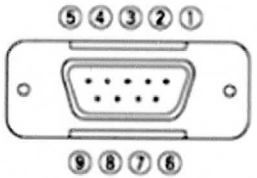
Icom OPC617 Accessory Cable for F1721/2821 and F621 radios
Connection Diagram



Install the OPC-617 as shown below.



Cut off the bushing as in the illustration, when you install the optional OPC-617.



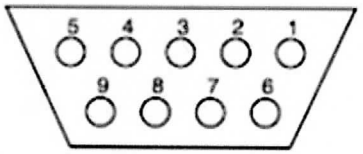
F1721 pinout

1. Dimmer cont. IN
2. AF OUT
3. Det. AF OUT
4. Mod. IN
5. PTT control IN or
6. Horn drive cont. OUT
7. AF GND
8. Det. AF GND
9. Mod. GND

F621 pinout

1. LCD backlit cont. IN
2. AF OUT
3. Det. AF OUT
4. Mod. IN
5. PTT control IN
6. Horn drive cont. OUT
7. AF GND
8. Det. AF GND
9. Mod. GND

DB-9, contact side of socket connector, or solder side of pin connector



Michael Wright, K6MFW
5-27-15

4-1-5 AF AMPLIFIER CIRCUIT (MAIN unit)

The AF amplifier circuit amplifies the demodulated AF signals to drive a speaker.

AF signals from the FM IF IC (IC1, pin 9) are amplified at the AF amplifier (IC7a) and then pass through the high-pass filter (IC6) whose characteristics are controlled by the "AFHPF" line. When "AFHPF" is at a high level, the cut off frequency is shifted higher to remove CTCSS or DTCS signals.

The filtered signals from IC6 (pin 14) pass through the AF switching IC (IC11, pin 8, 9) and are applied to the level controller (IC12, pin 16, 15). The audio level controlled signals are again passed through the AF switching IC (IC11, pin 2, 1), applied to the de-emphasis circuit (IC4) with frequency characteristics of –6 dB/octave, and are then power amplified at the AF amplifier (IC9) to drive a speaker.

4-1-6 RECEIVER MUTE CIRCUITS (MAIN and FRONT units)

• NOISE SQUELCH

The noise squelch circuit cuts out AF signals when no RF signals are received. By detecting noise components in the AF signals, the squelch circuit switches the AF mute switch.

A portion of the AF signals from the FM IF IC (IC1, pin 9) are applied to the level controller (IC12, pin 24). The level controlled noise components are output from pin 23 and are applied to the active filter in IC1 (pin 8). Noise components are amplified and output from pin 7 and are then applied to the noise detector section (pins 10, 11). The detected noise signals are rectified and output from pin 13 without smoothing.

The noise signal (NOIS) from IC1 (pin 13) is applied to the CPU (FRONT unit; IC1, pin 19). The CPU analyzes the noise condition and outputs the RMUT signal via the I/O expander IC (IC13) to toggle the AF mute switches (IC11a).

• CTCSS AND DTCS

The tone squelch circuit detects AF signals and opens the squelch only when receiving a signal containing a matching subaudible tone (CTCSS or DTCS). When tone squelch is in use, and a signal with a mismatched or no subaudible tone is received, the tone squelch circuit mutes the AF signals even when noise squelch is open.

A portion of the AF signals from the AF amplifier (IC7b) passes through the low-pass filter (FRONT unit; Q5) to remove AF (voice) signals and is applied to the CTCSS or DTCS decoder inside the CPU (FRONT unit; IC1, pin 97) via the "CTCIN" line to control the AF mute switch via the I/O expander IC (IC13).

4-2 TRANSMITTER CIRCUIT

4-2-1 MICROPHONE AMPLIFIER CIRCUIT (MAIN unit)

The microphone amplifier circuit amplifies the audio signals from the microphone, within +6 dB/octave pre-emphasis characteristics, to a level needed for the modulation circuit.

The AF signals from the microphone are amplified at the AF (IC8b) and limiter (IC8a) amplifiers and are then passed through the pre-emphasis circuit (R152, C180) which has +6 dB/octave pre-emphasis characteristics.

The pre-emphasized signals are passed through the splatter filter (IC7a) and are then applied to the AF switching IC (IC11, pins 11, 10).

The signals are applied to the level controller (IC12, pins 16, 15). The deviation level controlled signals are passed through the AF switching IC (IC11, pins 3, 4), and are then applied to modulation circuit as the "MOD" signal.

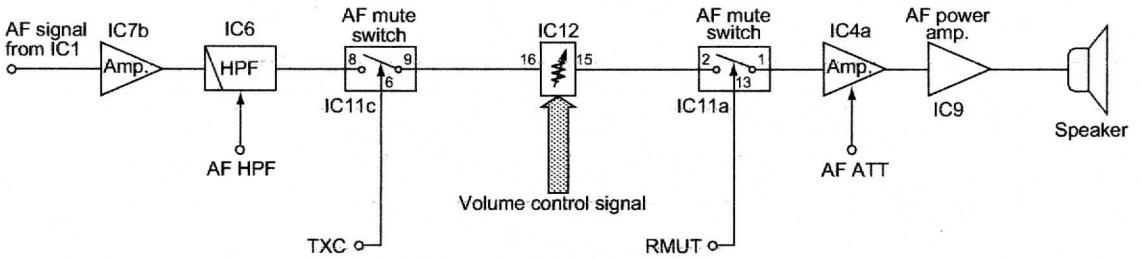
The narrow/wide switch (Q35) is connected to the input of the splatter filter (IC7a) and switched by the "NWC" signal coming from the I/O expander IC (IC13). When "NWC" is at a high level, the narrow/wide switch (Q35) shifts the filter cut-off frequency for narrow deviation selection.

4-2-2 MODULATION CIRCUIT (MAIN unit)

The modulation circuit modulates the VCO oscillating signal (RF signal) using the microphone AF signals.

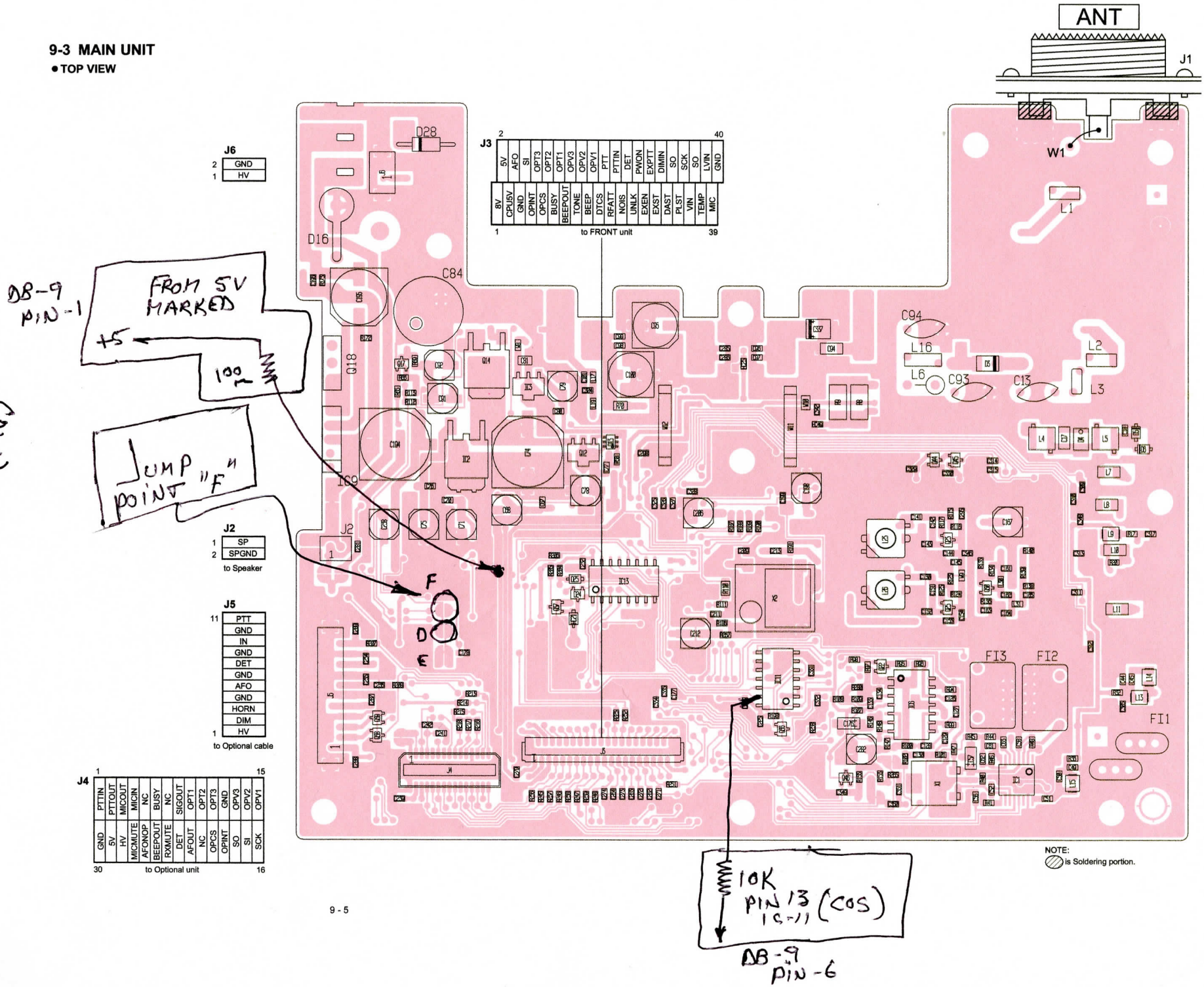
The "MOD" signals from the AF switching IC (IC11, pin 4) change the reactance of D21 to modulate the oscillated signal at the TX VCO circuit (Q25). The modulated signal is amplified at the buffer amplifiers (Q28, Q30) and is then applied to the drive amplifier circuit.

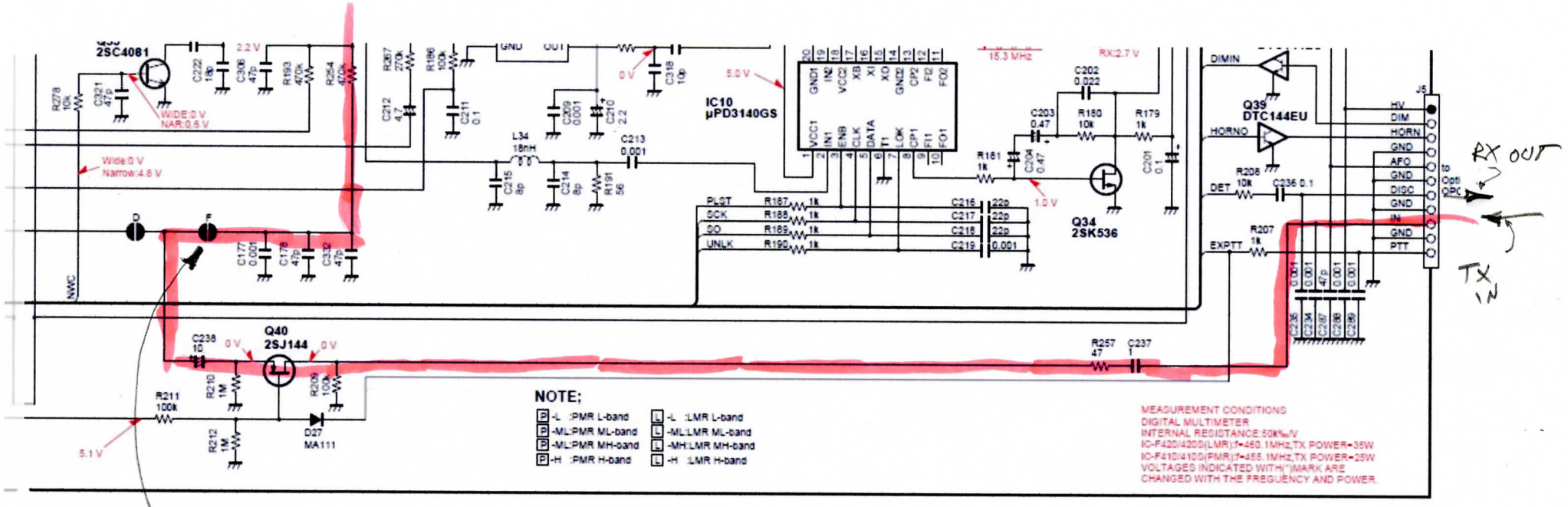
• AF circuit



9-3 MAIN UNIT

• TOP VIEW

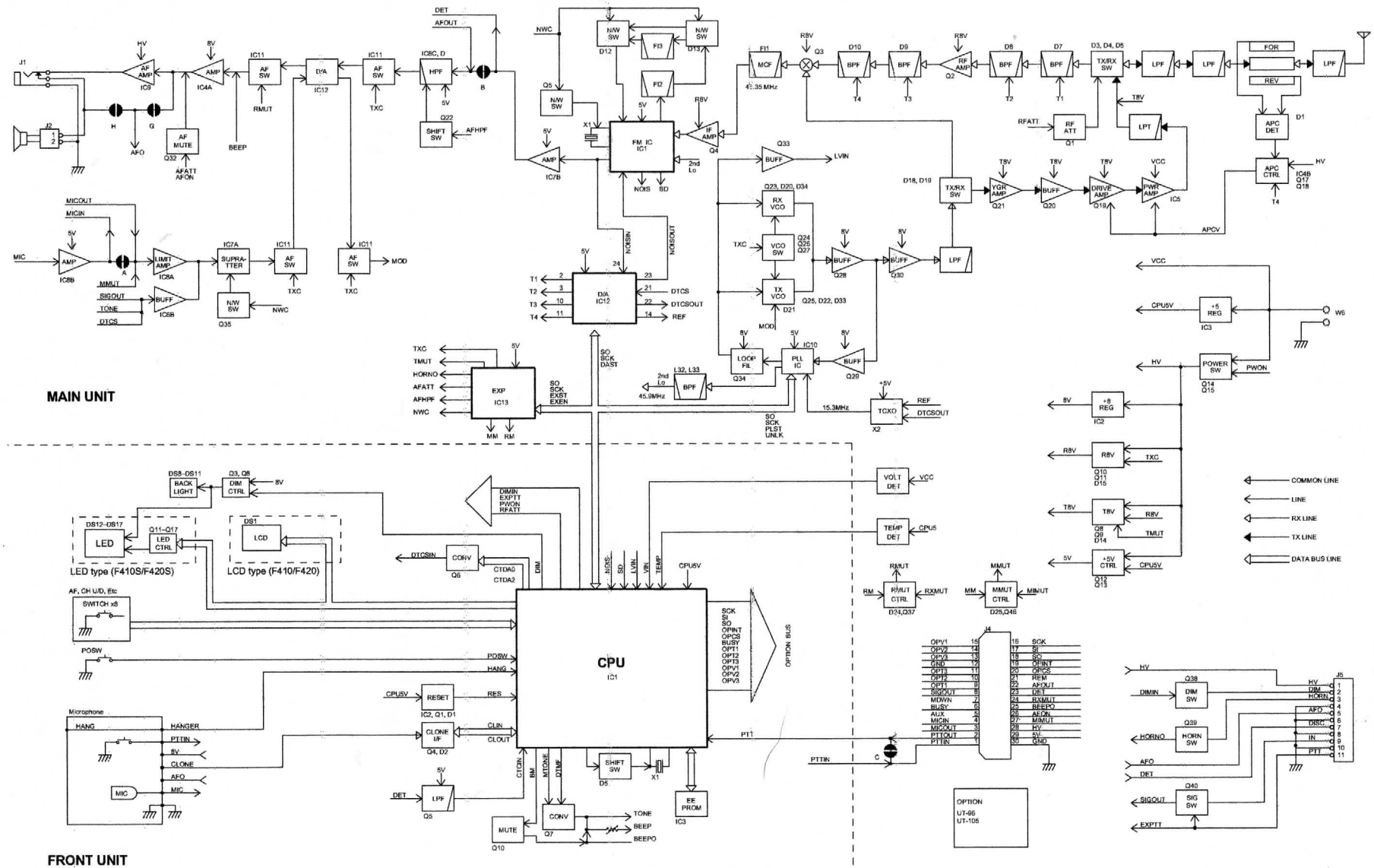




EXTERNAL TX AUDIO —
JUMP "F"

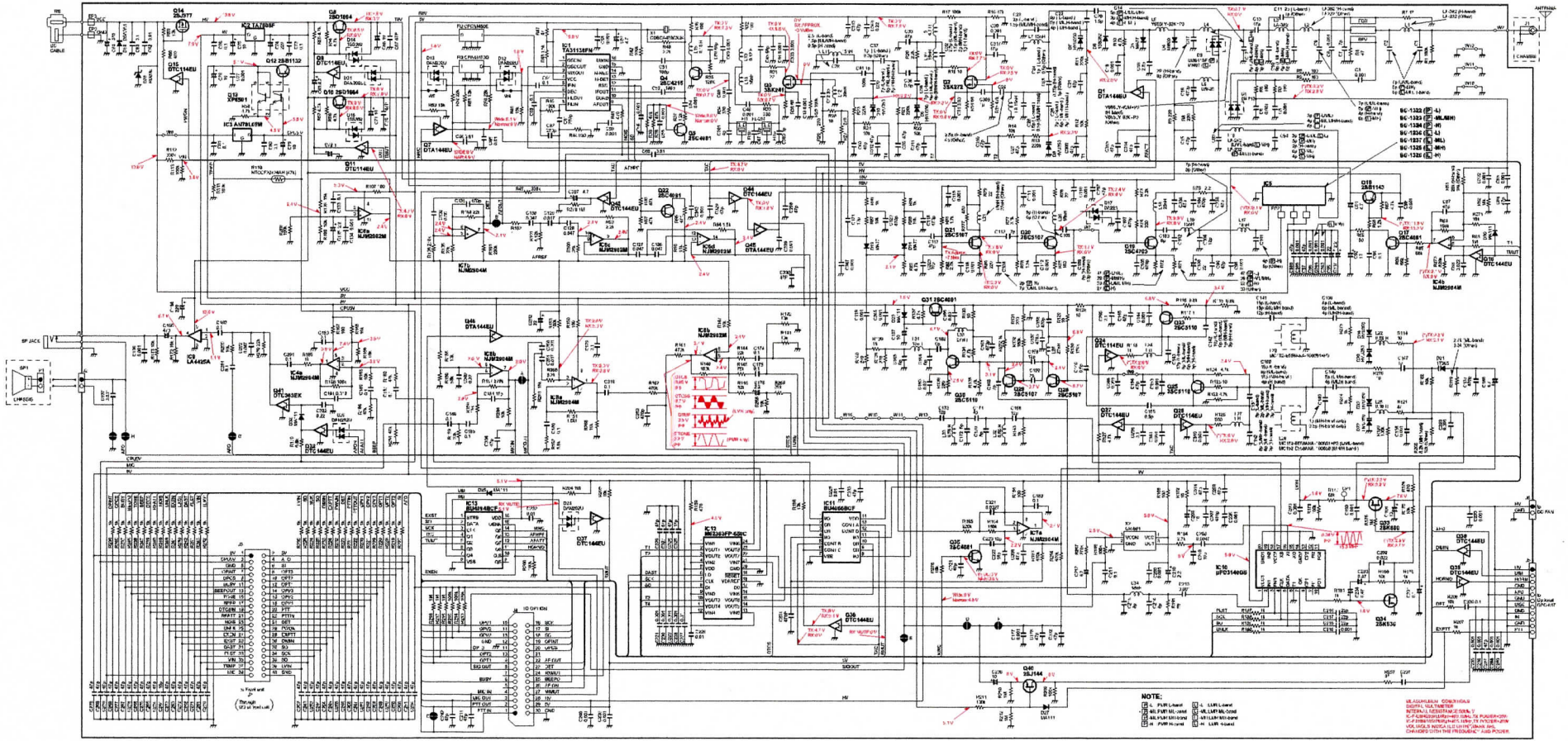
(22-1)

SECTION 10 BLOCK DIAGRAM



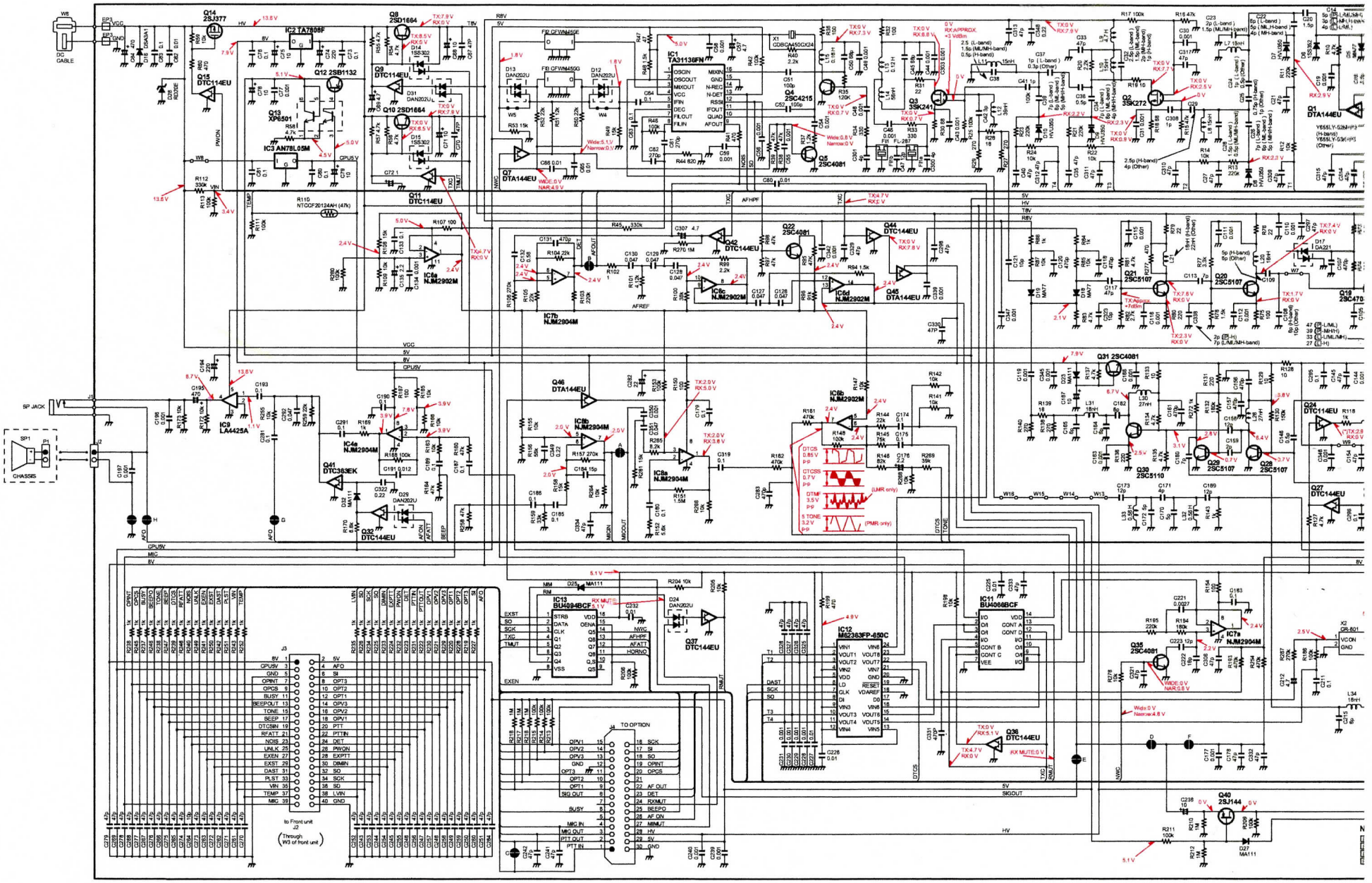
(93-1)

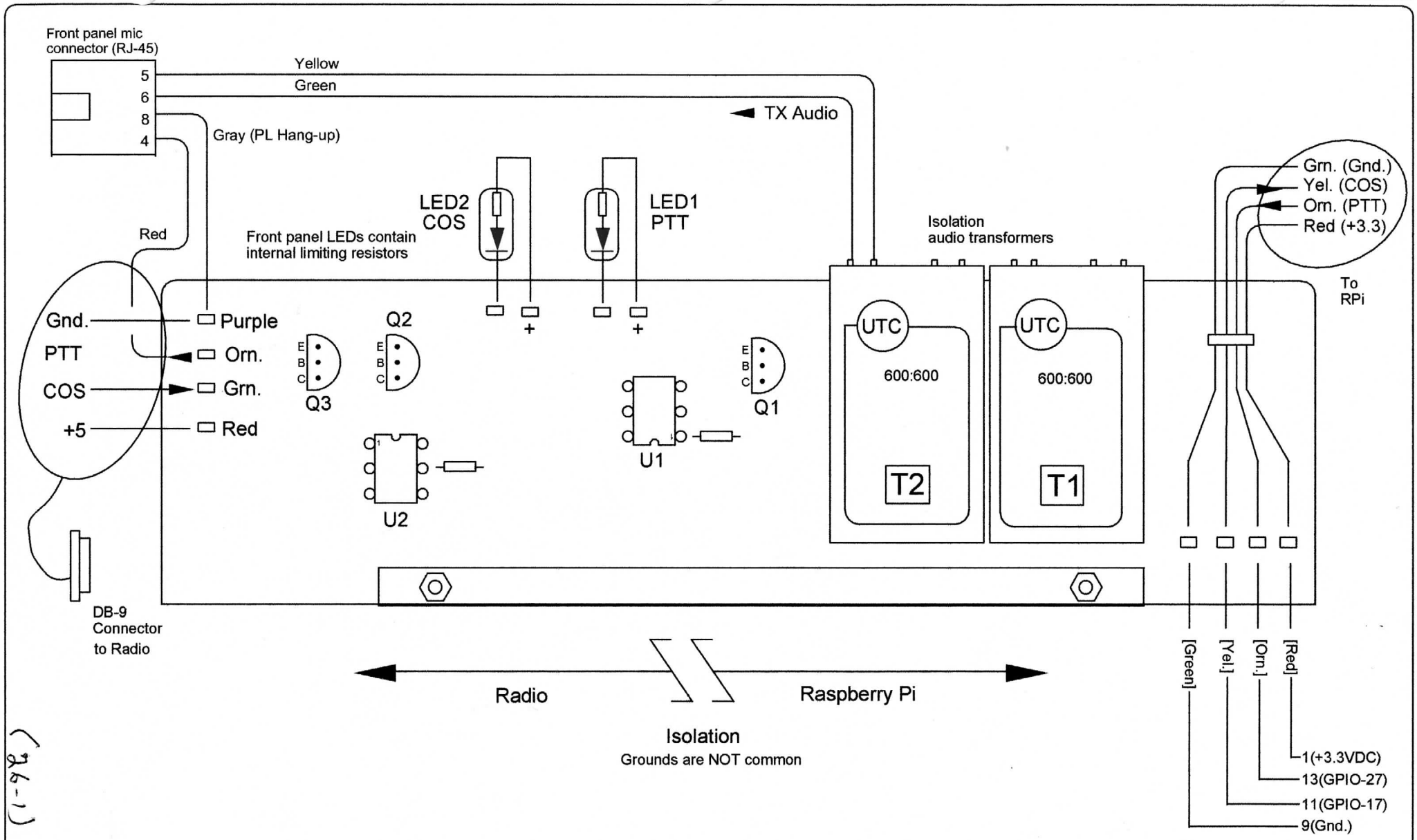
• MAIN UNIT



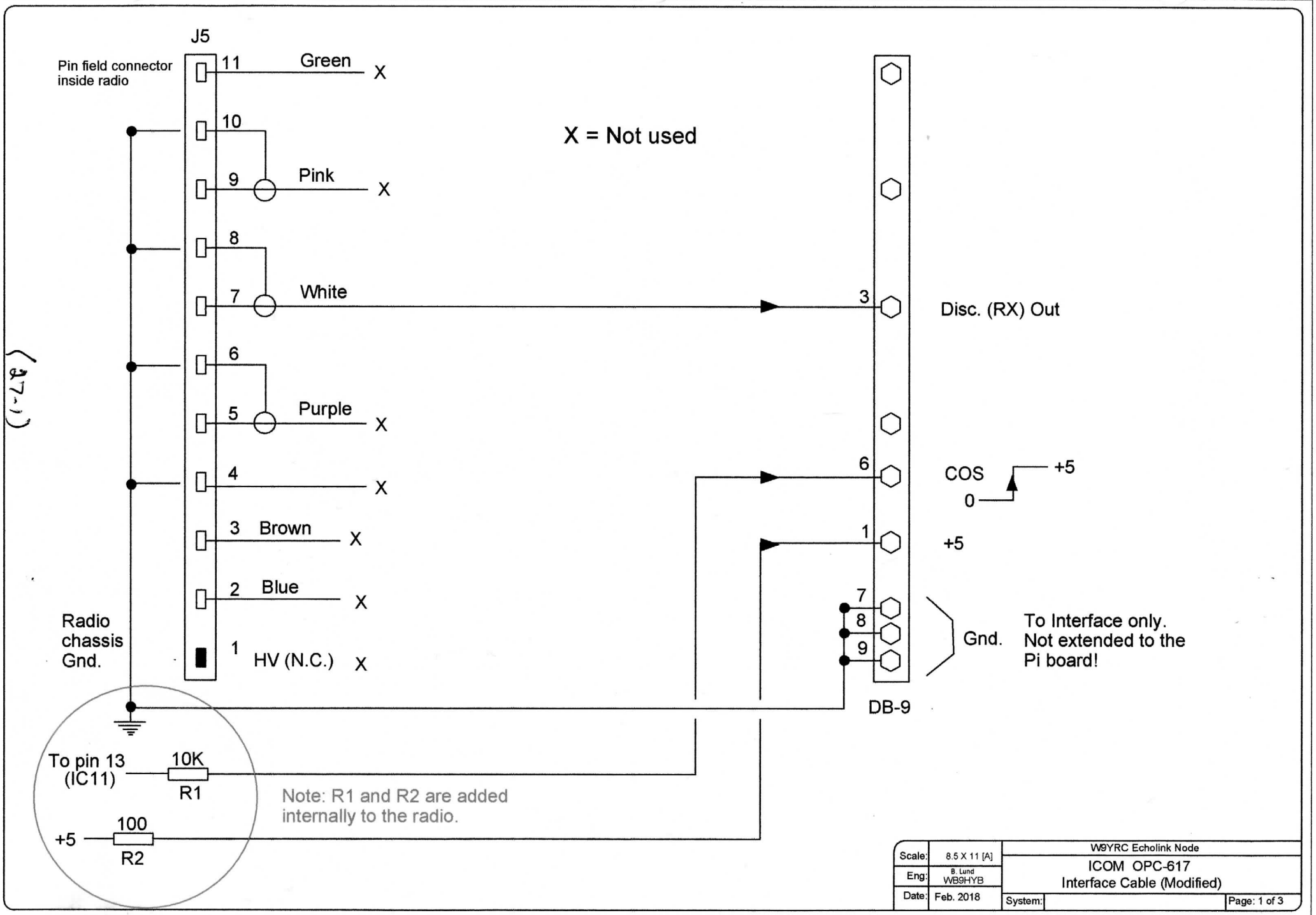
LEFT SIDE

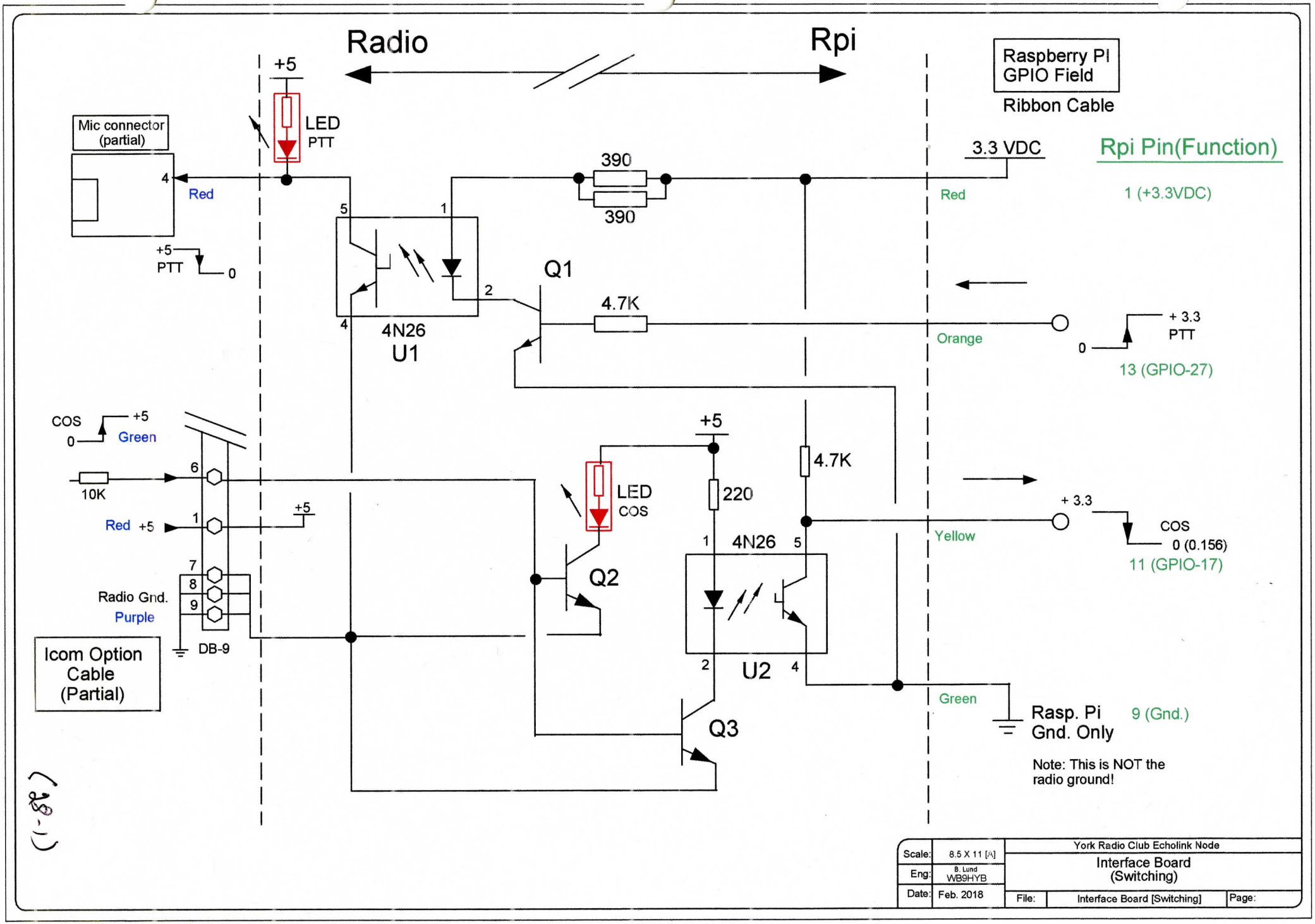
RIGHT SIDE

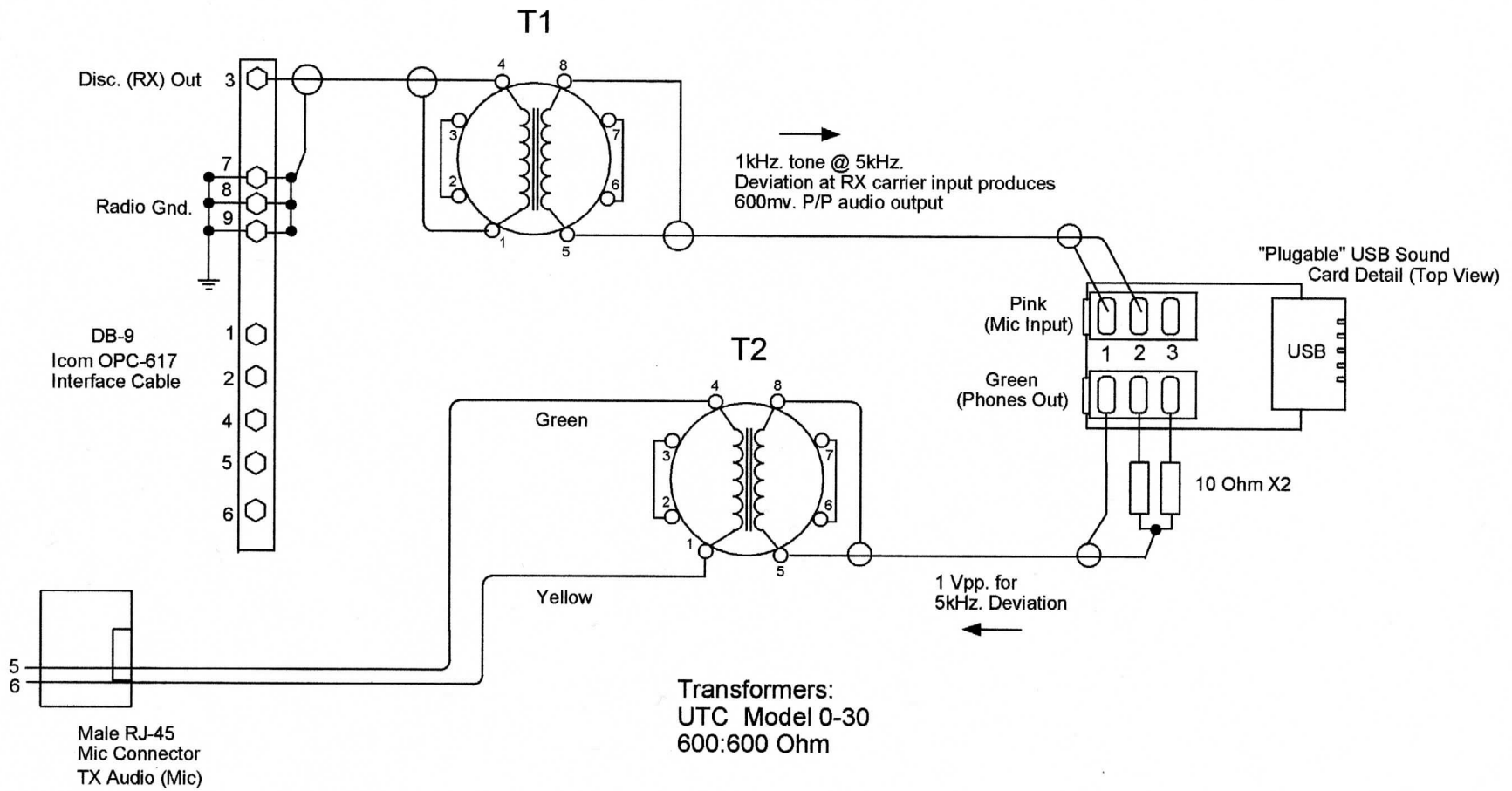




Scale:	8.5 X 11 [A]	York Radio Club Echolink Node	
Eng:	B. Lund WB9HYB	Interface Board Layout	
Date:	March 2018	File: Interface Board [Layout]	Page:

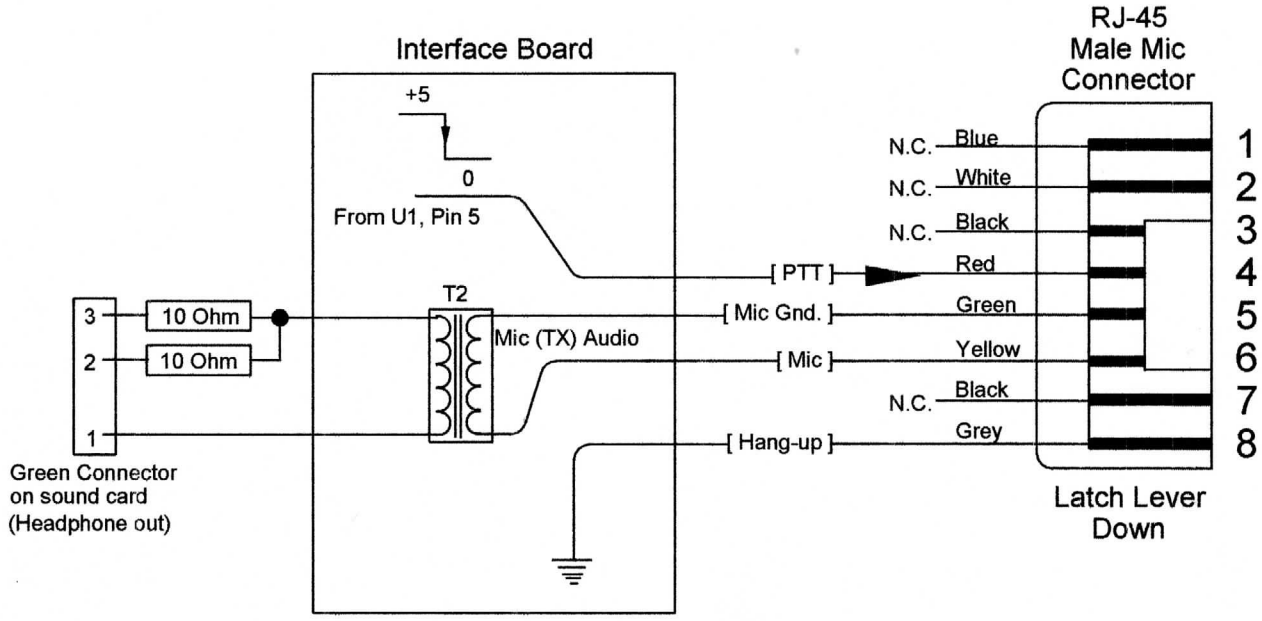

J5
Pin field connector inside radio
11 Green X
10
9 Pink X
8
7 White
6
5 Purple X
4 X
3 Brown X
2 Blue X
1 HV (N.C.) X
X = Not used
3 Disc. (RX) Out
6 COS +5
0
1 +5
7
8
9 Gnd.
To Interface only. Not extended to the Pi board!
DB-9
Radio chassis Gnd.
To pin 13 (IC11) 10K R1
+5 100 R2
Note: R1 and R2 are added internally to the radio.
(37-1)
Scale: 8.5 X 11 [A]
Eng: B. Lund WB9HYB
Date: Feb. 2018
W9YRC Echolink Node
ICOM OPC-617
Interface Cable (Modified)
System:
Page: 1 of 3


Radio
Rpi
Raspberry PI
GPIO Field
Ribbon Cable
Mic connector
(partial)
+5
LED
PTT
4
Red
+5
PTT
0
5
1
2
4
4N26
U1
Q1
390
390
4.7K
3.3 VDC
Red
Rpi Pin(Function)
1 (+3.3VDC)
Orange
+ 3.3
PTT
0
13 (GPIO-27)
COS
0
+5
Green
6
10K
Red +5
1
+5
7
8
9
Radio Gnd.
Purple
DB-9
Icom Option
Cable
(Partial)
LED
COS
Q2
+5
220
4.7K
1
4N26
5
2
U2
4
Q3
Yellow
+ 3.3
COS
0 (0.156)
11 (GPIO-17)
Green
Rasp. Pi
Gnd. Only
9 (Gnd.)
Note: This is NOT the
radio ground!
Scale: 8.5 X 11 [A]
Eng: B. Lund WB9HYB
Date: Feb. 2018
York Radio Club Echolink Node
Interface Board
(Switching)
File: Interface Board [Switching]
Page:

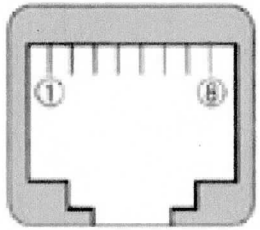


Scale:	8.5 X 11 [A]	York Radio Club Echolink Node	
Eng:	B. Lund WB9HYB	Interface Board (Audio)	
Date:	Feb. 2018	File: Interface Board [Audio].dwg	Page:

(39-1)

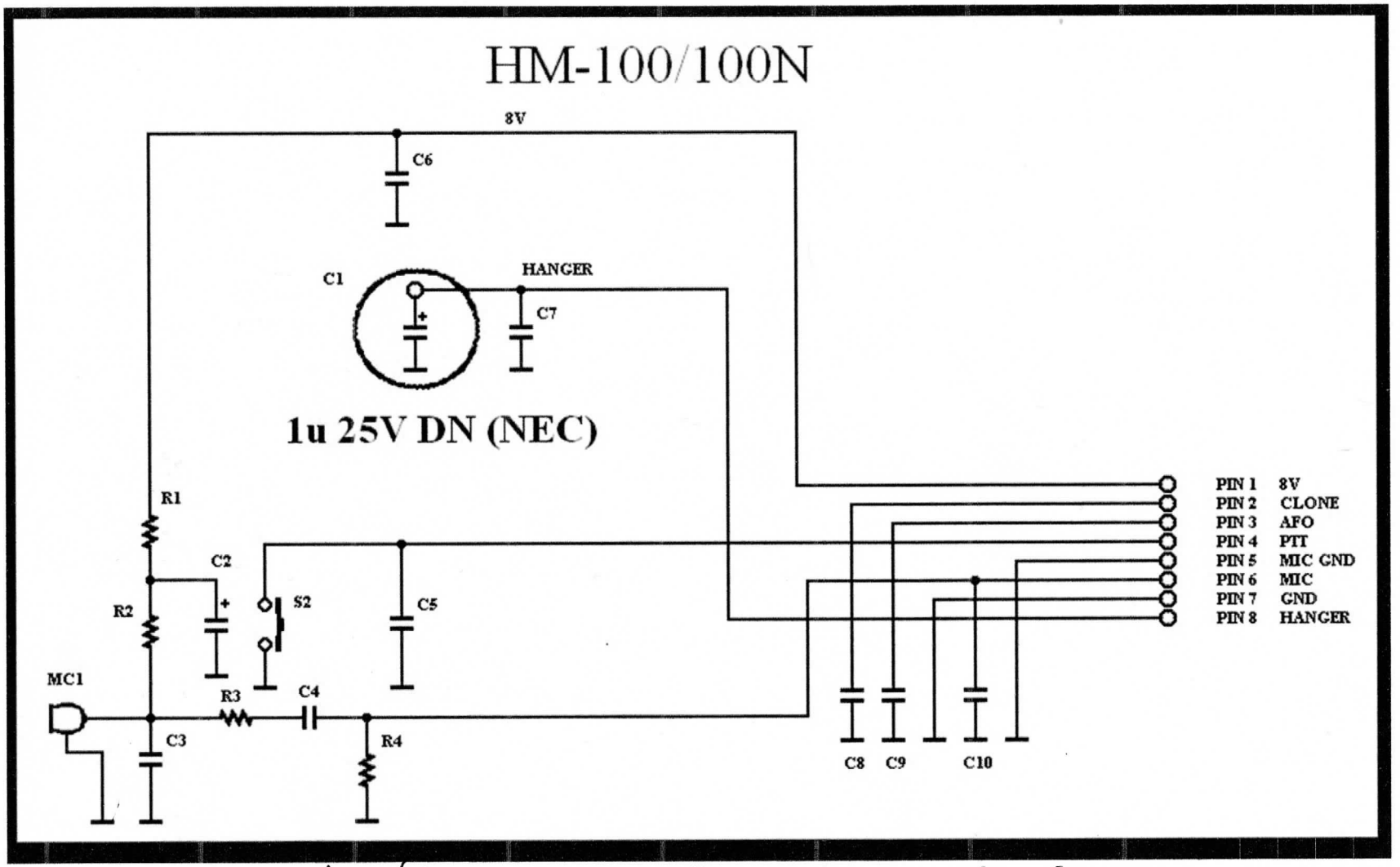


◇ **Microphone connector** (front panel view)



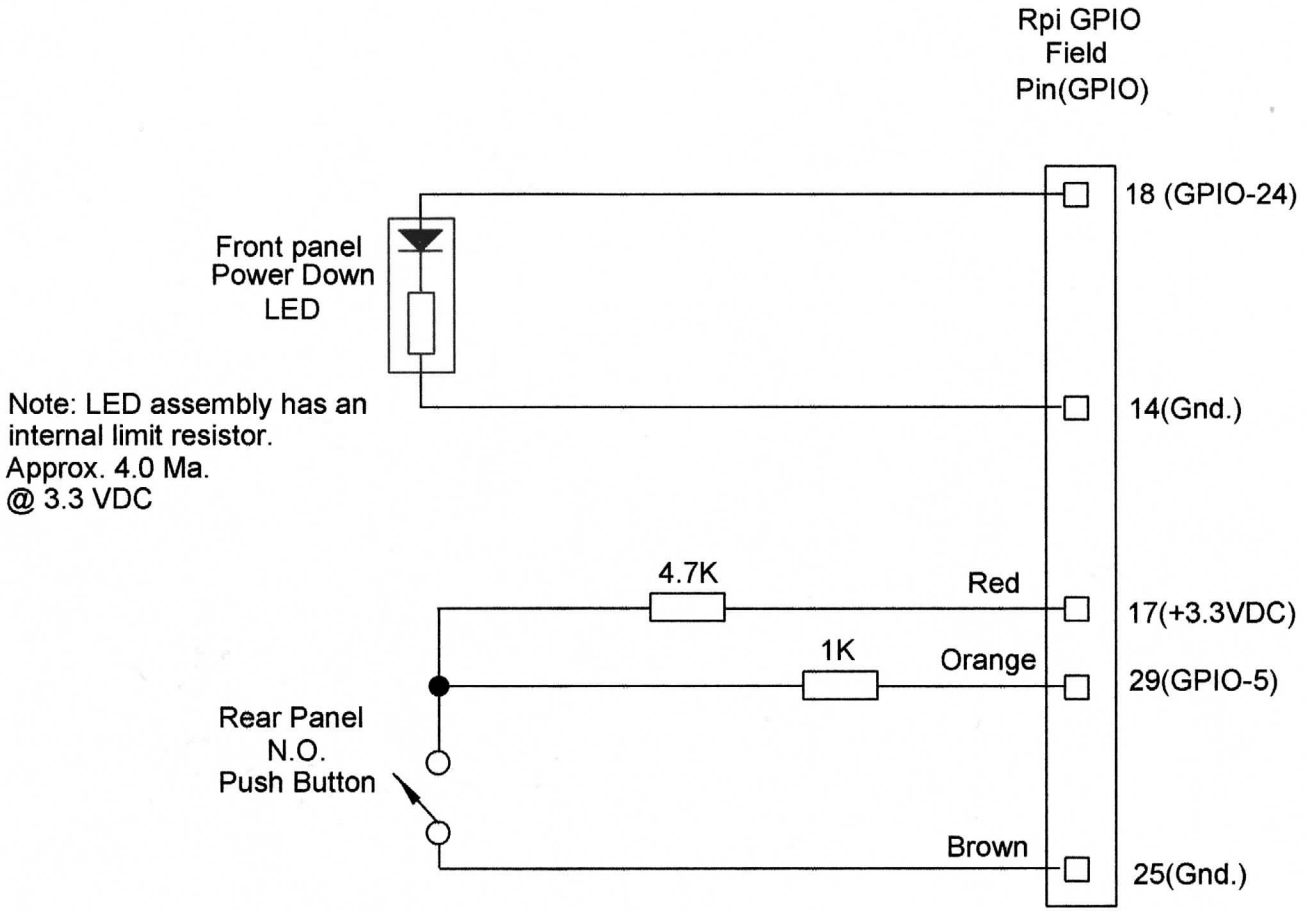
① +8 V DC output
② Frequency up/down
③ NC (no connection)
④ PTT
⑤ GND (microphone ground)
⑥ MIC (microphone input)
⑦ GND
⑧ NC (no connection)

Scale:	8.5 X 11 [A]	York Radio Club Echolink Node	
Eng:	B. Lund WB9HYB	Front Panel Mic Connector/Cable	
Date:	April 2018	File: Mic Cable and Connector	Page:



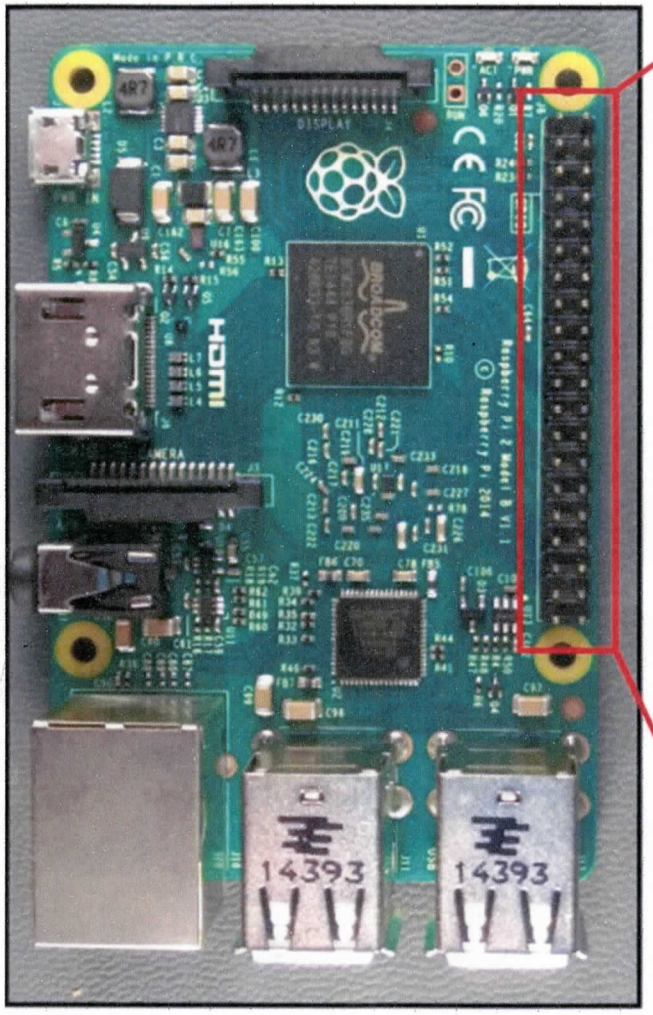
ICOM HAND MIC - (NOT USED) = SHOWS FRONT MIC CONNECTOR PIN-OUT

(1-13)

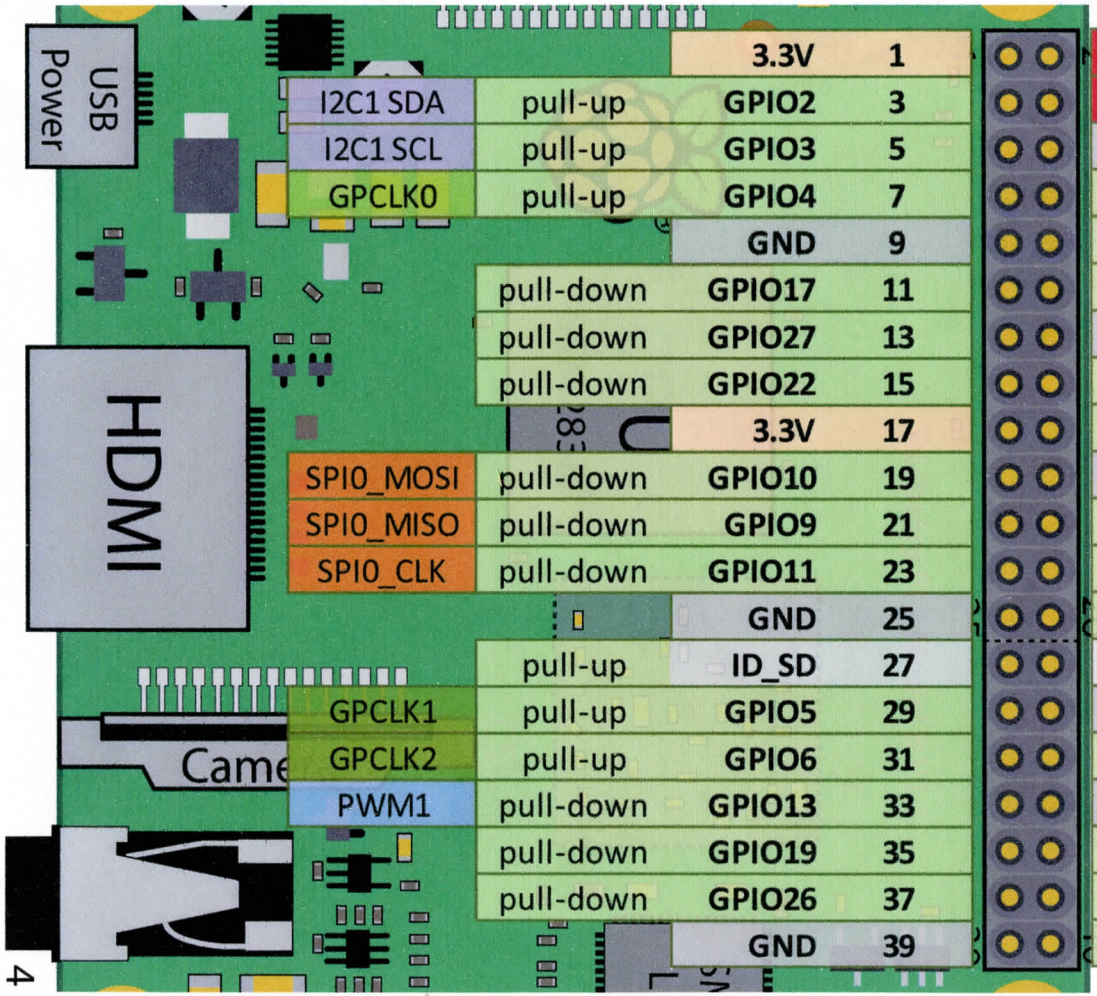


(38-1)

Scale:	8.5 X 11 [A]	York Radio Club Echolink Node	
Eng:	B. Lund WB9HYB	Power Down Button	
Date:	April 2018	File:	Power Down Button.dwg



Alternate Function							Alternate Function
	3.3V PWR	1		2	5V PWR		
I2C1 SDA	GPIO 2	3		4	5V PWR		
I2C1 SCL	GPIO 3	5		6	GND		
	GPIO 4	7		8	UART0 TX		
	GND	9		10	UART0 RX		
	GPIO 17	11		12	GPIO 18		
	GPIO 27	13		14	GND		
	GPIO 22	15		16	GPIO 23		
	3.3V PWR	17		18	GPIO 24		
SPI0 MOSI	GPIO 10	19		20	GND		
SPI0 MISO	GPIO 9	21		22	GPIO 25		
SPI0 SCLK	GPIO 11	23		24	GPIO 8		SPI0 CS0
	GND	25		26	GPIO 7		SPI0 CS1
	Reserved	27		28	Reserved		
	GPIO 5	29		30	GND		
	GPIO 6	31		32	GPIO 12		
	GPIO 13	33		34	GND		
SPI1 MISO	GPIO 19	35		36	GPIO 16		SPI1 CS0
	GPIO 26	37		38	GPIO 20		SPI1 MOSI
	GND	39		40	GPIO 21		SPI1 SCLK



		3.3V	1	2	5V			
I2C1 SDA	pull-up	GPIO2	3	4	5V			
I2C1 SCL	pull-up	GPIO3	5	6	GND			
GPCLK0	pull-up	GPIO4	7	8	GPIO14	pull-down	TXD0	
		GND	9	10	GPIO15	pull-down	RXD0	
	pull-down	GPIO17	11	12	GPIO18	pull-down	PWM0	
	pull-down	GPIO27	13	14	GND			
	pull-down	GPIO22	15	16	GPIO23	pull-down		
		3.3V	17	18	GPIO24	pull-down		
SPI0_MOSI	pull-down	GPIO10	19	20	GND			
SPI0_MISO	pull-down	GPIO9	21	22	GPIO25	pull-down		
SPI0_CLK	pull-down	GPIO11	23	24	GPIO8	pull-up	SPI_CE0_N	
		GND	25	26	GPIO7	pull-up	SPI_CE1_N	
	pull-up	ID_SD	27	28	ID_SC	pull-up		
GPCLK1	pull-up	GPIO5	29	30	GND			
GPCLK2	pull-up	GPIO6	31	32	GPIO12	pull-down	PWM0	
PWM1	pull-down	GPIO13	33	34	GND			
	pull-down	GPIO19	35	36	GPIO16	pull-down		
	pull-down	GPIO26	37	38	GPIO20	pull-down	GPCLK0	
		GND	39	40	GPIO21	pull-down	GPCLK1	

Misc. Software Notes And information

System Startup/Shutdown

Even though the Raspberry Pi computer appears as a small unassuming system, it is a very capable computing device sporting the Raspbian distribution of the Linux multi-tasking operating system and a dedicated version of the "SVXlink" program which is the heart of the Echolink system.

When shutting down almost all computing systems, the general practice is to execute a systematic shutdown procedure to assure that all software processes and memory accesses have completed and stabilized. Otherwise, simply "pulling-the-plug" may result in memory corruption or catastrophic software failure! Of course, a power company AC failure is always possible. I've had several power glitches and AC failures which didn't seem to have much effect other than rebooting the system. In any event, it is always good practice to do a systematic shut-down.

Anyway, starting up the system is completely automatic. On power-up, the system initializes all the software processes and then if successful, momentarily lights the far-right front panel LED. To make sure that all of the software processes have terminated, and any memory writing has successfully completed, a structured shutdown mechanism has been added to this system. A push button on the rear panel initiates a software routine that executes a structured sequence for shutting down the system. To initiate the shutdown, press the button on the back panel for about 2-seconds until the front panel LED fast flickers indicating the software shut-down has initiated. At this point, wait about 10-seconds for the shutdown to complete before removing AC power. The Pi and the Linux operating system has now terminated and will require an AC power cycle to restart.

Although a full description of the software shutdown process and the Linux SystemD services is beyond the scope of this document, it was thought that a brief discussion should be presented here because these items are not part of the distribution of the operating system or the Svxlink/Echolink application. In general, this shutdown mechanism consists of three (3) parts: The hardware button and LED indicator attached to GPIO pins on the Raspberry Pi, a short "C" program to watch for the button press and finally a Linux "SystemD Unit/process" installation. The button/LED hardware is detailed in the schematic section of this manual on page (32-1). The following page(s) detail the SystemD unit file and the short "C" program. The "C" program simply runs in an infinite loop waiting for the button press before executing a shut-down. A small delay is added to this loop to debounce the switch closure. A SystemD Unit file is called at boot-up which installs the "C" program as a runtime process in Linux.

- **SystemD Unit file:**
  Located in: /etc/systemd/system/
  File name: ShutdownButton.service

```
#=================================================
# This is the SystemD unit file for the rear panel
# shutdown push button. This is called at system start.
# Bob Lund - WB9HYB [1/14/2019]
#=================================================
[Unit]
Description=Push button system shutdown

[Service]
Type=simple
ExecStart=/home/pi/ShutdownButton

[Install]
WantedBy=multi-user.target
```

- **"C" Program:**
  Located in: /home/pi
  File name: ShutdownButton
  Note: Make sure that this compiled file is executable (green file name). If needed, change the file permission to executable by running:
  "chmod 755 ShutdownButton" to make it executable.

- **Useful Linux commands:**

- Top	To see running services
- Sudo systemctl status ShutdownButton	check process status

Note: Once the program (Compiled version of "ShutdownButton.c") and the unit file (ShutdownButton.service) are placed in the above directories, the following command is required to enable the process to run at boot-up and install as a permanent running process. This is only required once and, as of this writing the process is installed:

```
sudo systemctl enable ShutdownButton
```

```
/*=================================================================== *
*                          [Name: ShutdownButton]                     *
* =================================================================== *
```

A simple button for input as a system service which loops until the button is pushed then shuts down the system and exits. This is installed as a SystemD service.

```
* ------------------------------------------------------------------
*                        B. Lund - WB9HYB [1/13/2019]
*
*  Compile with: gcc -Wall ShutdownButton.c -o ShutdownButton -lwiringPi
*
*      Test run with: ./ShutdownButton&
* ================================================================*/

#include <wiringPi.h>
#define LED 5       // LED on pin 5
#define Button 21   // Button on pin 21. Active pulls pin low

int main (void)
{
int count=0;
int i=0;

      wiringPiSetup();
      pinMode(LED,OUTPUT);
      pinMode(Button,INPUT);

      /*************************************************************
      * We'll start by just turning on the LED for a moment during *
      *  system power-up. This will indicate that this service has *
      *  installed.                                                *
      *************************************************************/
            digitalWrite(LED,HIGH);
            delay(1500);
            digitalWrite(LED,LOW);

      for(;;) // Endless loop
            {
                  count=0;

                  while(digitalRead(Button) == 0)
                        {
                              ++count;
                              delay(10);

                              if(count > 200)
                              {
                                    for(i=0;i<25;++i)
                                    {
                                    digitalWrite(LED,HIGH);
                                    delay(50);
                                    digitalWrite(LED,LOW);
                                    delay(50);
                                    }
                                    system("sudo shutdown -h now");
                              }
                        }
            }

      return 0;
}
```

Updating the Raspberry

Tip: This step is absolutely necessary. Type the commands at the prompt, and wait for each of the commands to complete. There is no need to expand the operating system in the MicroSD Card, as Stretch does this on first boot.

1. sudo raspi-config to open the configuration menu then at option 5 enable VNC
2. At option 7 Advanced options, select A5 for Mode 16
3. Exit raspi-config, but do not reboot yet.
4. At the prompt type sudo apt update
5. At the prompt type sudo apt-get upgrade
6. When the commands have all finished type sudo reboot. Exit the SSH app.

Sound Card Configuration

1. Re-open RealVNC viewer to the raspberry pi.
2. Re-open a terminal screen and type the following command at the prompt.
3. sudo alsamixer
4. you should now see that your USB sound card is the default sound card. type cntrl-f5 to view all the sliders.
5. set the first slider to around 60% the second to zero, the third to a maximum of 12 and in the fourth type M to mute the amplifier.
6. Type escape then type
7. sudo alsactl store This will save these settings even after re-booting.
8. sudo adduser svxlink (This will prepare the card for the installation of SVXLink.)
9. Make the password svxlink, as you won't need to use this any further.

How to Clone Your Raspberry Pi SD Card

Cloning the SD card is simple. Just follow these steps:

1. Get everything set up just the way you want it on your Raspberry Pi, whatever you're using it for. Then shut down the Pi and remove the SD card. Insert the SD card into your computer.
2. Start up Win32DiskImager, a program that you probably have from when you first set up your Pi. (If you're on OS X or Linux, you'll have to use the dd command as described here instead of these steps).
3. In the "Image File" box, enter the path of your soon-to-be image file. For example, I put mine in `C:\Users\Whitson\images\myraspbmc.img`
4. Under the "Device" box, select your SD card.
5. Click the "Read" button to create the image file from your card.
6. When it's done creating the image file, you can eject your SD card and put it back in your Raspberry Pi. Keep that IMG file in a safe place.

Now, if anything ever goes wrong with your Pi, you can restore your fully-set-up image using the reverse instructions:

1. Insert the SD card back into your computer.
2. Head to the start menu or screen and type "disk management." Open the disk management program and find your SD card in the list.
3. Right-click and delete all the partitions on your SD card. When it's empty, right-click on it and format it (it doesn't matter what filesystem you format it to, your computer just needs to recognize it).
4. Open Win32DiskImager again and browse for your image file. Select your device from the Device dropdown just as you did before.
5. This time, click "Write" to write the image to the SD card.
6. When it finishes, eject the SD card and re-insert it into your Raspberry Pi. When you boot it up, it should be in the exact same state it was in when you first cloned the SD card.

Once you've done this, setting up your Pi from scratch will be a whole lot simpler!

How to back up your Raspberry Pi's SD card on Windows

Step 1: Open Win32 Disk Imager

You may recognize this program from our guide to installing Raspbian. It's our go-to for SD card writing and re-writing, and we'll use it here, too. If you don't have it yet, you can download it here.

Step 2: Set the drive and destination folder

In Win32 Disk Imager, use the drop-down menu labeled Device to choose the drive that corresponds to your SD card. Then click the little button with the folder on it and select the folder that you want your backup to end up in – but don't hit Open quite yet. Type a new filename into the File name: box – this will be the file name of your backup. The file type will be .img by default, which is good, because that's what you want. You can go ahead and hit Open now.

Step 3: Write the file

Now just hit the Read button – which should be easy enough to do, especially since Win32 Disk Imager will gray out the Write button when it sees that the .img file you've chosen doesn't exist (yet). Your work here is done! Win32 Disk Imager will do the rest. Eject your SD card when it's all done and keep enjoying your Pi with a little extra peace of mind.

Restoring your Raspberry Pi on Windows

If that grim day comes that you have to use this little backup, you'll find the process looks very familiar. Once again, you'll put an SD card in and select the proper drive from the drop-down window. Once again, you'll select your .img file – only this time, of course, it will already exist! And since it already exists, you'll see that the Write button is a viable option. That's exactly what you should hit, of course, and once you do Win32 Disk Imager will replace whatever is on your SD card with the backup disk image. Easy!

SD Card Preservation

Once you get the system working, it is advisable to create a backup of the card. There are many ways to do this. I just used a simple *dd* command to dump the SD card to a raw image file.

I found that after a year of continuous use, my 4 GB class 4 SD card became corrupted. I had initially suspected that the card was fried, but using f3 (a Linux alternative to h2testw) to check the card, it was reported fine. I was able to restore the node easily and in minutes with the DD image of the card.

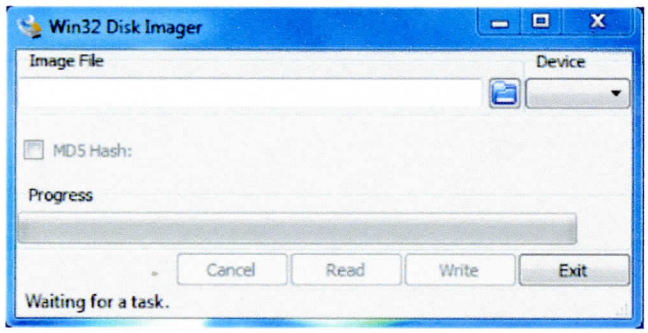
It is also common knowledge that you should have an SD card a few times bigger than you need to allow the wear leveling space to work. To aid this, I turned swap off on the Raspberry Pi:

```
sudo swapoff -a
sudo update-rc.d -f dphys-swapfile remove
sudo rm /var/swap
```

I also turned down the verbosity of the Apache web server and make SvxLink log into a RAM based filesystem, using tmpfs. On the Pi, */run/shm* is mounted in RAM and has a default size of 47 MB (on my Pi. 256MB model B). You could also consider running the Pi with the SD card in "read only" mode. Search the Pi forums for "read only" for more.

Finally, always be sure to power the Raspberry Pi down correctly – do not just pull the power – this causes file system errors.

Writing IMG File to the SD Card on Windows (Win32DiskImager)



1. Download and install the Win32DiskImager utility (if you haven't already).
2. Obtain the latest copy of the disk image for the project from our Downloads page.
3. Extract the image file from the downloaded .ZIP file, so you now have "distribution-name.img".
4. Insert the SD card into your SD card reader verify it's drive letter.
5. Run the Win32DiskImager (Depending on your computer and os , you many need to run with Administrator rights.)
6. Select the image file you extracted above.
7. Select the drive letter of the SD card in the device box. WARNING: Make sure to select the correct drive! You could inadvertently damage your hard drive if you are not careful.
8. Click Write and wait for the write to complete.
9. Exit Win32DiskImager and eject the SD card.

Last Updated: 2 years ago in Getting Started

1 0

Setting up the GPIO ports

We need to setup the GPIO ports. The simplest way is to add the following lines to the file rc.local.

1. sudo nano /etc/rc.local

```
#################
#GPIO SCRIPT #
#TO BE INSERTED#
#ON START-UP #
#################
# GPIO 17 as PTT to TX2
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction
sudo chmod 777 /sys/class/gpio/gpio17/value
# GPIO 18 as Squelch to RX2
echo 18 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio18/direction
sudo chmod 777 /sys/class/gpio/gpio18/value
#GPIO 24 as PTT to TX1
echo 24 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio24/direction
sudo chmod 777 /sys/class/gpio/gpio24/value
# GPIO 23 as Squelch to RX1
echo 23 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio23/direction
sudo chmod 777 /sys/class/gpio/gpio23/value
#end of GPIO Section
#Remove # at start of next line for permanent reboot on power loss.
#sudo svxlink –daemon –logfile=/var/log/svxlink.log
These lines go immediately before the final
exit 0
Save and exit the file.
```

GPIO Configuration

Whatever we decide to do with this installation, it will need an interface. There are so many different methods but we will concentrate on the most useful, that of using the General Purpose Input Output or GPIO. We will address two pins of the GPIO strip on the Raspberry so that one becomes a PTT driver, and the other becomes a squelch or COS/COR detector. We must remember that the GPIO is restrictive in that 3V is the maximum voltage, so we must buffer these suitably to drive our devices. However to enable the GPIO we have to set up a script in order to do so. We will use the on-board text editor to do this.

We type sudo nano /etc/rc.local where we modify a hidden file rc.local in the home directory /etc.

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
printf "My IP address is %s\n" "$_IP"
fi
################
#GPIO SCRIPT #
#TO BE INSERTED#
#ON START-UP #
################

# GPIO 17 as PTT to TX2
echo 17 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio17/direction
sudo chmod 777 /sys/class/gpio/gpio17/value
# GPIO 18 as Squelch to RX2
echo 18 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio18/direction
sudo chmod 777 /sys/class/gpio/gpio18/value
```

```
#GPIO 24 as PTT to TX1
echo 24 > /sys/class/gpio/export
echo out > /sys/class/gpio/gpio24/direction
sudo chmod 777 /sys/class/gpio/gpio24/value
# GPIO 23 as Squelch to RX1
echo 23 > /sys/class/gpio/export
echo in > /sys/class/gpio/gpio23/direction
sudo chmod 777 /sys/class/gpio/gpio23/value

#end of GPIO
#Remove # at start of next line for permanent reboot on power loss.
#sudo svxlink –daemon –logfile=/var/log/svxlink.log
exit 0
```

This is a "Bash" script and needs a little explanation. Firstly #!/bin/bash at the beginning of the file designates it as a script. We will ignore the rest and start at the GPIO section.

This script has been written for driving two transceiver pairs with the same logic, thus giving the possibility of a UHF and VHF repeater working in double-duplex. The # character is a comment line (except for the first line). However we shall concentrate on GPIO 23 and 24, with GPIO 23 as the COS squelch and GPIO 24 as the PTT driver. "echo" simple writes a value to a variable and stores it, and sudo chmod 777 gives the program full permission to use it. This file is a script so has to be enabled as executable, so in nano we ctrl-o to save, then ctrl-x to exit nano, then sudo chmod +x /etc/rc.local for this to occur.

The line second for bottom needs editing later to remove the # to provide the svxlink service, but we leave it for now until it is fully tested.

Sound Configuration

I use a plug in USB sound card, as they are cheap and convenient, and they can be used for two sets of transceivers at the same time, by using the left and right channels. The first thing that has to be done is to turn off or blacklist the internal sound port on the Raspberry. So we create a file as follows: sudo nano /etc/modprobe.d/alsa-blacklist.conf and in the editor type the line "blacklist snd_bcm2835" (without the quotes) and ctrl-o to save the file and ctrl-x to exit the editor. now type sudo reboot. When the Raspberry returns we log back in and type sudo alsamixer. We now have a graphical representation of the sound controller and hopefully the USB Sound card listed in the header. Ideally we need to type ctrl-f5 to open all options. We should set the speaker audio around 55-60 by using the keyboard arrow keys. The next box (Mic) should be set at zero, and the next box (also Mic but Capture) set low to begin with, and Auto Gain Off. Now we type escape to exit the controller and the following command sudo alsactl store to lock these settings against change in the event of a restart.

Further changes to levels may need to be accomplished in the interface.

sm0svx / svxlink

InstallBinRaspbian

Gunnar edited this page 8 hours ago · 10 revisions

Installation instruction for Raspbian

This instruction explains how to install SvxLink with packages from the Raspbian official software repository. It does not cover all parts of the installation and configuration in detail, but is supposed to be easy to understand with limited knowledge and experience of Linux.

Note: Please browse through the instruction quickly, before you begin.

Download and install image

Download the image, Raspbian Stretch: https://www.raspberrypi.org/downloads/raspbian/

Details: *The official Raspbian installation instruction can be found here. https://www.raspberrypi.org/documentation/installation/installing-images/README.md*

Download the image installation program **"Etcher"** and install the image on the card: https://etcher.io/

Before inserting the SD-card in the Raspberry, **enable ssh**. Add a file named "ssh" in the root of the SD-card. https://www.raspberrypi.org/documentation/remote-access/ssh/

Insert the SD-card and boot the Raspberry. The Raspberry PI will start with DHCP enabled (automatically request an ip-address from your local router). Find the ip-address of the raspberry pi, and logon. ***Details:*** *https://www.raspberrypi.org/documentation/remote-access/ip-address.md*

Login and password

If you have a monitor and keyboard connected to your Raspberry, just logon at the prompt after it has started. You can also connect via ssh, without a monitor and keyboard connected. Good tools for SSH on Windows are: Putty or MobaXterm

username: pi

password: raspberry

Change the default password "raspberry" to something else, the command passwd will change the password:

```
passwd
```

Set static ip-address

Optionally set a static ip address by editing the dhcpcd.conf file. See the example in the bottom of the file. You can use "nano" editor. To save and exit the editor, press ctrl-x and answer Y to save:

```
sudo nano /etc/dhcpcd.conf
```

```
# Example static IP configuration:
interface eth0
static ip_address=192.168.0.10/24
```

▸ Pages 51

Home
Feature List
Project News
Screenshots
Installation Instructions
Qtel User Docs
SvxLink Server User Docs
Developer Documentation
Getting Support
External Links
System Examples
Config Examples
Contributing

Clone this wiki locally
https://github.com/sm0svx/s
Clone in Desktop

```
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.0.1
static domain_name_servers=192.168.0.1 8.8.8.8
```

Now is a good time to reboot your Raspberry.

```
sudo reboot
```

If you can not reconnect to the new static ip address, something is wrong. Try power off/on, or connect a monitor.

Install SvxLink

The most current version of the SvxLink packages for Raspbian is called **17.12.1-1**. The packages is not available directly in in the Raspbian Stretch image, but in the "testing" software repository (Buster release). To install this release, we temporarily add the Buster software repository and install SvxLink.

```
echo 'deb http://mirrordirector.raspbian.org/raspbian/ buster main' | sudo tee /etc/apt/sou
sudo apt-get update
```

Note: Do NOT upgrade your complete system. We only want the SvxLink applications from the "buster" release!

Now install the svxlink-server package. The apt-get command will automatically select other packages needed for SvxLink to run.

```
sudo apt-get -t buster install svxlink-server
```

Now remove the buster repository again.

```
sudo rm /etc/apt/sources.list.d/svxlink.list
```

Install SvxLink sounds

Next install the English sounds for SvxLink with the following commands.

```
cd /usr/share/svxlink/sounds/
sudo wget https://github.com/sm0svx/svxlink-sounds-en_US-heather/releases/download/14.08/sv
sudo tar xvjf svxlink-sounds-en_US-heather-16k-13.12.tar.bz2
sudo ln -s en_US-heather-16k en_US
```

Basic test of the SvxLink Installation

It is now time to do some basic tests of the SvxLink installation. Do NOT connect any extra USB-sound card, just connect a speaker to the 3.5mm audio output. The Speaker will be our TX. Since the Raspberry does not have a Microphone input, we need to disable the default RX (RX=NONE).

Edit the main SvxLink configuration file:

```
sudo nano /etc/svxlink/svxlink.conf
```

In the SimplexLogic section, change the RX to NONE, and set your CALLSIGN.

```
[SimplexLogic]
TYPE=Simplex
RX=NONE
TX=Tx1
MODULES=ModuleHelp,ModuleParrot,ModuleEchoLink,ModuleTclVoiceMail
CALLSIGN=SK3W
```

SvxLink is normally started from the a start script, but as a test it can be started on the command prompt. We start the program as the svxlink user. You will see some errors on missing configuration for echolink, this can be configured later.

```
sudo -u svxlink svxlink
```

```
pi@raspberrypi:/usr/share/svxlink/sounds $ sudo -u svxlink svxlink
SvxLink v1.5.99.17 Copyright (C) 2003-2017 Tobias Blomberg / SM0SVX
SvxLink comes with ABSOLUTELY NO WARRANTY. This is free software, and you are
welcome to redistribute it in accordance with the terms and conditions in the
GNU GPL (General Public License) version 2 or later.
Using configuration file: /etc/svxlink/svxlink.conf
--- Using sample rate 48000Hz
Starting logic: SimplexLogic
Loading RX: NONE
Loading TX: Tx1
Loading module "ModuleHelp" into logic "SimplexLogic"
      Found /usr/lib/arm-linux-gnueabihf/svxlink/ModuleHelp.so
      Module Help v1.0.0 starting...
Loading module "ModuleParrot" into logic "SimplexLogic"
      Found /usr/lib/arm-linux-gnueabihf/svxlink/ModuleParrot.so
      Module Parrot v1.1.1 starting...
Loading module "ModuleEchoLink" into logic "SimplexLogic"
      Found /usr/lib/arm-linux-gnueabihf/svxlink/ModuleEchoLink.so
      Module EchoLink v1.3.99.1 starting...
*** ERROR: Please set the EchoLink callsign (ModuleEchoLink/CALLSIGN) to a real callsign
*** ERROR: Initialization failed for module ModuleEchoLink in logic SimplexLogic
Loading module "ModuleTclVoiceMail" into logic "SimplexLogic"
       Found /usr/lib/arm-linux-gnueabihf/svxlink/ModuleTcl.so
      Module Tcl v1.0.1 starting...
SimplexLogic: Event handler script successfully loaded.
```

Enter *# on the keyboard, and you should hear your SimplexLogic identify itself in the speakers.

```
SimplexLogic: digit=*
SimplexLogic: digit=#
Tx1: Turning the transmitter ON
Tx1: Turning the transmitter OFF
```

Stop SvxLink by pressing ctrl-c. If the volume is too low, use alsamixer to increase the volume.

Configuration

Congratulations! You have successfully installed SvxLink! Now the fun begins.

You should read the Post installation stuff chapter.

Good to know

The normal way to start and stop SvxLink on Raspbian is via scripts.

```
sudo systemctl start svxlink
```

```
sudo systemctl stop svxlink
```

Monitor the system via the svxlink log.

```
tail -f /var/log/svxlink
```

All manuals are available from the command prompt.

```
man svxlink
man svxlink.conf
```

Custom SVXLink Configuration

Warning: This is an advanced topic. You will need to have a decent understanding of the Linux command line and file system, how SVXLink it's self works and is configured, as well as understanding of the PHP programming language. If you don't feel comfortable with these areas, then do not proceed with these instructions.

So you want to make ORP do more that what it can out of the box and you are OK with doing some coding? Perhaps you have specialized needs that are not built into ORP yet or are mor niche to your setup. This article will help point you in the right direction. If won't go into what those SVXLink configurations are, but rather it will point you to where to make the modifications.

Some Background First

Firstly, it is important to understand how OpenRepeater is structured and how it works. If you don't already know the OpenRepeater Project combines a number of elements into a prepackaged setup to run on single board computers like the Raspberry Pi. These elements include the base OS (Debian), the SVXLink program which is our core logic, a web server and c components that make the ORP Web Front end function. This web front end its part of the ORP project that helps with configuring some of the most common setting for the basic operati a repeater.

When you log into the web front end and make changes they will immediately get saved into a database. At this point no changes are made to the SVXLink configuration. You will notice a prompt appears at the top of the page prompting you to "rebuild & restart". When changes are made, a flag is set to make this prompt appear. Once you are done making changes, you c click on the "rebuild & restart" button. What this does is runs a PHP build script that reads all the settings from the database and performs some conditional logic and writes out multiple configuration files including the main SVXLink configuration file, some other configuration and override files (like custom TCL logics) to modify the behavior of features such as identificat & courtesy tones, GPIO setting to initialize the GPIO pins with the OS, and then it will restart SVXLink to read in the new configurations.

The main SVXLink configuration file is located here:

```
/etc/openrepeater/svxlink/svxlink.conf
```

This is an example of what that code might look like.

```
[GLOBAL]
MODULE_PATH=/usr/lib/arm-linux-gnueabihf/svxlink
LOGICS=RepeaterLogic
CFG_DIR=svxlink.d
TIMESTAMP_FORMAT="%c"
CARD_SAMPLE_RATE=16000
#LOCATION_INFO=LocationInfo
#LINKS=LinkToR4

[RepeaterLogic]
TYPE=Repeater
RX=Rx1
TX=Tx1
MODULES=ModuleHelp,ModuleParrot
CALLSIGN=A0AAA
SHORT_IDENT_INTERVAL=1
LONG_IDENT_INTERVAL=5
EVENT_HANDLER=/usr/share/svxlink/events.tcl
DEFAULT_LANG=en_US
RGR_SOUND_DELAY=1
REPORT_CTCSS=
TX_CTCSS=ALWAYS
MACROS=Macros
FX_GAIN_NORMAL=0
FX_GAIN_LOW=-12
IDLE_TIMEOUT=1
OPEN_ON_SQL=1
OPEN_SQL_FLANK=OPEN
IDLE_SOUND_INTERVAL=0

[Rx1]
TYPE=Local
AUDIO_DEV=alsa:plughw:0
AUDIO_CHANNEL=0
SQL_DET=GPIO
GPIO_SQL_PIN=gpio27
SQL_HANGTIME=10
SQL_START_DELAY=1
SQL_DELAY=10
SIGLEV_SLOPE=1
SIGLEV_OFFSET=0
SIGLEV_OPEN_THRESH=30
SIGLEV_CLOSE_THRESH=10
DEEMPHASIS=1
```

If you want to see what your file looks like after you makes changes you can CAT the contents of the file with the following command:

```
cat /etc/openrepeater/svxlink/svxlink.conf
```

This uses some of the most common settings and sections required to run a repeater. This is the main file that SVXLink will read in upon startup or restarting. To see what additional feat that the SVXLink core program supports, view the SVXLink manpages.

Where to Make the Modifications

As you can probably figure out by reading above, you can certain make changes to the SVXLink.conf file and other config files that are written to by ORP, but if you make changes within t web interface and "rebuild & restart" then those changes are going to be over written. The way around this is to modify the svxlink_update.php file or it's included files.

The SVXLink_update.php file is located here:

```
/var/www/openrepeater/functions/svxlink_update.php
```

At The Top of the File...

You will see where the files are called in that read from the database.

```
// Get Settings from SQLite
include_once("../includes/get_settings.php");

// Get Port Settings from SQLite
include_once("../includes/get_ports.php");
```

At The Bottom of the File...

Where it writes out the configuration files.

```
/* WRITE CONFIGURATION &amp; TCL FILES */

file_put_contents('/etc/openrepeater/svxlink/svxlink.conf', $svx_global . $svx_repeaterLogic . $svx_ports);
if (isset($moduleEchoLink)) { file_put_contents('/etc/openrepeater/svxlink/svxlink.d/ModuleEchoLink.conf', $moduleEchoLink); }
file_put_contents('/etc/openrepeater/svxlink/local-events.d/CustomLogic.tcl', $tclOverride);
file_put_contents('/etc/openrepeater/svxlink/svxlink_gpio.conf', $gpioConfigFile);
```

...and clears the flag and restarts SVXLink...

```
/* CLEAR SETTINGS UPDATE FLAG TO CLEAR BANNER AT TOP OF PAGE */
$memcache_obj = new Memcache;
$memcache_obj-&gt;connect('localhost', 11211);
$memcache_obj-&gt;set('update_settings_flag', 0, false, 0);

$shellout = shell_exec('sudo /usr/bin/openrepeater_svxlink_restart');
```

All the Magic Happens in the Middle...

Here is an example of one section of code. The example sets some EchoLink settings.

```
/* MODULE: ECHOLINK CONFIGURATION SETTINGS */
```

It goes without saying...make a backup copy of this file before you start making changes to it. As you can see, SVXLink expects things grouped into sections. Some of these settings are ha coded (which you could change here if you'd like) and other have the variables inserted. In some cases some additional PHP logic is added, or things are grouped into functions, or extern files are called to help keep things more organized. Knowing this, it should give you a good understanding of how to reverse engineer this for your own needs. It's probably best to make changes at a time. When done, save the file and go to the web gui, initiate a change to invoke the "Rebuild & Restart" bar. When you run that this file will be called. If all goes well, you changes will be applied to the SVXLink configuration file and be loaded. If you see a white screen or an error message, then that is usually a good indicator that you have improperly formatted code.

Happy Modding!

Last Updated: 2 years ago in Advanced / Command Line



PE2KMV

Contact About PE2KMV Museum Shack Projects Frequencies Links

Knowledge Base

SVXLink hassle

Posted on July 23, 2016 | By Ronald Bouwens

For a friend of mine, also HAM, I wanted to build a turnkey Echolink node. The necessary hardware was a matter of combining a few ideas, collecting parts and one afternoon of drilling and sawing. Except the interface also a Raspberry Pi was part of the deal.

The software side had some more challenges. There were a couple of conditions:

- no hassle with displays, keyboards and strange commands, so;
- starting the necessary application during boot phase;
- connecting via an Echolink proxy;
- autoconnecting to a predefined node;
- enabling commands via DTMF.

With this in mind I started with 'TheLinkBox'. All of this is well documented, but I ran into some issues. I was unable to setup autoconnection and DTMF was nagging as well. And because time was running out I've tried my luck with SVXLink. But also here some surprised showed up. Perhaps other radio amateurs can use the wheels I've (re)invented.

Follow me!

Follow

Actuele weer in Valkenburg (L)

Base system

It all began with with Raspbian. How to create a bootable SD card is well documented on the internet, so we'll skip that part. For the first boot of Raspbian a mouse and keyboard are needed!

After having Raspbian up and running, mouse and keyboard can be detached and the Echolink interface can be connected. Be aware that the interface as described activates TX when SVXLink is not running / not configured. So keep the transceiver off for now!

Log in to the RPI with the standard credentials (assuming the password hasn't been changed yet):

Username = pi

Password = raspberry

First run all updates. Keep in mind this can take a while.

sudo apt-get update

sudo apt-get upgrade

sudo apt-get dist-upgrade

reboot

HINT: the user 'pi' is a standard user without 'root' access. Therefore prior to all commands 'sudo' has to be used. Instead you can once type '*sudo su*', which grants root access for the rest of the session. Be aware that use of root access disables all safeties so you're free to mess up your system. In that case you'll have to start over with a fresh image on an SD card.

Now it's time for the ALSA module. That module is not be installed in the standard setup of Raspbian.

sudo apt-get install alsa-tools alsa-oss

sudo modprobe snd-pcm-oss

sudo modprobe snd-mixer-oss

If this went well, two audio devices are available. Check this with the command

*ls -l /dev/dsp**

This will bring up two audio devices:

9°C|°F
Humidity: 64%
Pressure: 1005.54 hPa
Wind: 9 km/h E
Temperature
Wind
Sat, 24 19:00
Sat, 24 22:00
9° 9°
6° 6°
OpenWeatherMap

Recent Posts

- DMR thoughts
- Motorola MTM800: what's in a name?
- SMS functions via DMR
- DAPNET POCSAG Transmitter
- First 2m simplex QSO from the KWA tunnel

Search ... Search

/dev/dsp

/dev/dsp1

To be sure some modules are included in the boot phase, the file '/etc/modules' has to be edited. Use an editor like 'nano', 'vi' or 'mc'. Personally I prefer 'nano'. How to use this editor is not part of this guideline. The internet offers enough documentation on that topic.

sudo nano -w /etc/modules

Now add two lines:

snd-mixer-oss

snd-pcm-oss

After saving the file and closing the editor it's time to reboot the RPi. It's good to have a display still connected. Although the text on the verbose screen runs way to fast to read, you can easily see whether or not an error shows up during boot. Those lines are flagged with the word 'Failed' in red.

sudo reboot

Finally the USB soundcard (in the Echolink interface) needs to be the default one. First open the file '/lib/modprobe.d /aliasses.conf' with your favorite editor (in this example again 'nano')

sudo nano /lib/modprobe.d/aliases.conf

Now search for the line 'options snd-usb-audio index=-2'. Then change '-2' into '0', so it reads:

options snd-usb-audio index=0

Save '/lib/modprobe.d/aliasses.conf' and close the editor. Now it's time for the file '/boot/config.txt'.

sudo nano /boot/config.txt

Search for the line dtparam=audio=on and put a hash '#' in front of it, so it looks like:

#dtparam=audio=on

Save the file '/boot/config.txt' and close the editor. Now again the RPi has to be rebooted. Watch the output on the screen: as soon as red text appears something went wrong during the previous steps.

sudo reboot

Afterward a bunch of packages has to be installed.

sudo apt-get -y install subversion libsigc++-2.0-dev g++ make libsigc++-1.2-dev libgsm1-dev libpopt-dev tcl8.5-dev libgcrypt-dev libspeex-dev libasound2-dev alsa-utils cmake libopus-dev qt4-dev-tools librtlsdr-dev doxygen

Installing SVXLink

First of all we'll need to add the user 'svxlink' to the system and assign it to the group 'daemon':

useradd -G daemon svxlink

Now it's time to get hold of 'SVXLink' and have it installed. There are multiple guidelines how to get this from a repository, but I couldn't get one of those links running. Just downloading from www.svxlink.org proved to be the fastest way.

Unpack the downloaded file with the command

tar xvfz [bestandsnaam].tar.gz

Using the 'ls' command shows a new directory with the name 'svxlink' followed by a release number. Enter this directory and than continue to the subdirectory 'src'.

cd svxlink[nummer]/src

In this subdirectory you'll have to create another subdirectory called 'build', and enter it.

mkdir build

cd build

First some things have to be arranged and generated before compiling can start. The how and why is not part of this guideline.

cmake -DCMAKE_INSTALL_PREFIX=/usr -DSYSCONF_INSTALL_DIR=/etc \ -DLOCAL_STATE_DIR=/var ..

Now compiling can start with the commands below (in exactly the provided order). This will take some time again.

make

make doc

make install

ldconfig

Configuring SVXLink

Now the fun part can start by configuring and personalizing SVXLink. Start with the global configuration in the file '/etc/svxlink/svxlink.conf'.

sudo nano /etc/svxlink/svxlink.conf

Here some adjustments have to be made. To make things easy I've added the sections where you can find everything.

[SimplexLogic]

CALLSIGN=[eigen call]-L

[Rx1]

#SERIAL_PORT=/dev/ttyUSB0

[Tx1]

AUDIO_DEV=alsa:hw:0

PTT_PORT=/dev/ttyUSB0

HINT: depending on your configuration you can search for the CTCSS settings. When using CTCSS you can enter the correct subtone here. It will be announced then in the beacon text.

Save '/etc/svxlink/svxlink.conf' and close the editor. Now it's time to configure the Echolink module:

sudo nano /etc/svxlink/svxlink.d/ModuleEcholink.conf

Also this file needs some adjustments.

[ModuleEchoLink]

SERVER=europe.echolink.org

CALLSIGN=[eigen call]-L

PASSWORD=[echolink wachtwoord]

SYSOPNAME=[eigen naam]

LOCATION=[QTH en QRG]

USE_GSM_ONLY=1

PROXY_SERVER=[proxy server naar keuze]

PROXY_PORT=8100

PROXY_PASSWORD=PUBLIC

LINK_IDLE_TIMEOUT=0

AUTOCON_ECHOLINK_ID=[node nummer waarmee standaard verbonden dient te worden]

Save the file '/etc/svxlink/svxlink.d/ModuleEcholink.conf' and close the editor.

Installing voice response

Audio samples can be added to SVXLink. To do so first the audiofiles are to be downloaded. Download the 16k samples, because the new versions of SVXLink won't eat the 8k samples. These samples are to be unpacked to the directory '/usr/share/svxlink/sounds/en_US':

mkdir /usr/share/svxlink/sounds/en_US

After copying the samples to the directory shown above, SVXLink can be started again and the transceiver can be switched on again. SVXLink will send a spoken beacon text.

First tests

It's time now to boot the node for the first time.

sudo svxlink

The command above will provide output to your terminal window so you can observe what SVXLink is doing. When SVXLink correctly launches, the TX led of the interface will be off. Now you can switch on your transceiver.

At the Echolink website you can check whether or not the node is online.

If the node is online, use the DTMF keyboard of another (portable) transceiver to connect your node with the Echolink testnode by sending 4 times '9' followed by the hash '#'.

9999#

This will trigger a voice response, confirming you've successfully to the Echolink testnode. This node acts as a 'parrot', repeating the received audio a few seconds after the squelch has closed. Ideal to adjust your audio settings.

If this test has been successfully completed the transceiver

has to be switched off again. SVXLink has to be stopped by typing 'control' + 'C'.

[Ctrl][c]

Launch SVXLink during boot phase

SVXLink can be lauched using '/etc/rc.local'. You can do this by editing the file with the command

sudo nano /etc/rc.local

Just above the line 'exit 0' the line below has to be added:

/usr/bin/svxlink –daemon

When you reboot the RPI, SVXLink won't lauch automatically. When you manually trigger the script '/etc/rc.local' the node application will start as expected. It caused some headache to solve this. In the file '/etc/rc.local' two extra lines are to be added before '/usr/bin/svxlink –daemon'.

HOME=/home/pi

export HOME

Finally the file will look like this:

HOME=/home/pi

export HOME

/usr/bin/svxlink –daemon

exit 0

Finally save the file '/etc/rc.local' and close the editor. Now SVXLink will automatically launch after a reboot. This can easily be checked with the TX led of the interface. Now the transceiver can be powered up again. The node is ready to use.

Category: Echolink

Copyright 2018 Amateur Radio Page

sm0svx / svxlink

Branch: master ▾ svxlink / src / svxlink / modules / echolink / **ModuleEchoLink.conf** Find file Copy path

sm0svx ModuleEchoLink: Configurable remote roger sound b171ffa on Jan 21

1 contributor

42 lines (40 sloc) 1009 Bytes

```
[ModuleEchoLink]
NAME=EchoLink
ID=2
TIMEOUT=60
#ALLOW_IP=192.168.1.0/24
#DROP_INCOMING=^()$
#REJECT_INCOMING=^()$
#ACCEPT_INCOMING=^(.*)$
#REJECT_OUTGOING=^()$
#ACCEPT_OUTGOING=^(.*)$
#REJECT_CONF=0
#CHECK_NR_CONNECTS=2,300,120
SERVERS=servers.echolink.org
CALLSIGN=MYCALL-L
PASSWORD=MyPass
SYSOPNAME=MyName
LOCATION=[Svx] Fq, MyTown
#PROXY_SERVER=the.proxy.server
#PROXY_PORT=8100
#PROXY_PASSWORD=PUBLIC
#BIND_ADDR=10.20.30.40
MAX_QSOS=10
MAX_CONNECTIONS=11
LINK_IDLE_TIMEOUT=300
#AUTOCON_ECHOLINK_ID=9999
#AUTOCON_TIME=1200
#USE_GSM_ONLY=1
#DEFAULT_LANG=en_US
#COMMAND_PTY=/dev/shm/echolink_ctrl
#REMOTE_RGR_SOUND=0
DESCRIPTION="You have connected to a SvxLink node,\n"
            "a voice services system for Linux with EchoLink\n"
            "support.\n"
            "Check out http://svxlink.sf.net/ for more info\n"
            "\n"
            "QTH:     My_QTH\n"
            "QRG:     Simplex link on ???.??? MHz\n"
            "CTCSS:   My_CTCSS_fq_if_any Hz\n"
            "Trx:     My_transceiver_type\n"
            "Antenna: My_antenna_brand/type/model\n"

```