ICARC Raspberry PI FOX Transmitter

William Robison

April 29, 2020

This is the start of a manual for the ICARC Raspberry PI FOX Transmitter

It is a work-in-progress right now so suggestion for updates may be sent to **kc0jfq@n952.ooguy.com**.

Full size documents may be found here: http://n952.ooguy.com/eagle

http://n952.ooguy.com//home/wtr/eagle/JUICE/A102_73161/FOX_ICARC.tex

Contents

List of Figures

1 Glossary of Terms

There is an attempt being made to use some terms in this document in a precise manner. Some of the discussions become a bit muddled when terms are used casually.

2 Motivation

Why use Raspberry PI?

2.0.1 Audio

Because it has a souncard capability, we can transmit arbitrary audio clips.

See the TV series: Red Dwarf: "Talkie Taster".

2.0.2 Linux

The Raspberry-PI make sus of the Linux operating system. The opearation of the transmitter can be described using a simple shell script along with a few routines to manage the hardware interface.

2.1 Requirements

These are required for operation.

2.1.1 2M Band

We are using 2M handheld tranceivers as a primary detector.

We must stay within our licensed band.

2.1.2 Battery Operation

We must be able to carry multiple FOX Transmitters to the site of the hunt. AC power becomes extremely difficult.

2.1.3 Code Storage

We must store enough bits to identify the transmitter by sending CW or voice.

2.2 Desirements

These are desired features

2.2.1 Multiple frequency

We would like to be able to operate the transmitter on more than one frequency.

Implementation:
Clock Synthesizer (ICS525) clock synthesizer controlled by the zNEO processor. Any frequency that can ge generated by the ICS525 given its clock input may be selected.

2.2.2 Large CW tables

We get bored with the same old message over and over.

We would also like to be able to operate at any word rate.

Implementation:
Messqage traffic stored in Ferro-magnetic Random Access Memory (FRAM), currently 64Kb (8KB). Most of this memory can be used to store message traffic. CW chipping rate can be selected from 1-WPM to 50-WPM.

2.2.3 Easy Synchronization

There should be a method of synchronizing multiple transmitters.

This is not to imply that the transmitters must be in contact with each other.

Implementation:
Network communications port to allow multiople units to be connected to achieve clock synchronization.

2.2.4 Long Battery Life

This is to say we should be able to operate for the entire length of the fox hunt without having to replace batteries.

Implementation:
Case has room for a 6 cell AAA battery pack. This should allow for 12 to 24 hours of operation.

3 Theory of Operation

The heart of the RF system is a Renasas ICS307 frequency synthesizer. This device is programmed through an SPI-like serial interfaced and producesa CMOS logic level clock. The ICS307 produces three output channelsclocks, two of which are buffered and presented to a daughterboard for amplification. The daughterboard returns the amplified RF signal to the main board where it is run through a low pass filter and then out to the antenna.

The ICS307 clock is FM modulated by changing the capacitance on the the reference clock oscilaltor, pushing the carrier about the crystals nominal frequency. The modulation signal is provided by the Raspberry-PI.

The ICS307 is controlled by a Raspberry-PI Zero. The ICS307 setup bits are calculated by the PI and sent to the ICS307 over the PIs SPI interface. Discrete control bits come from PI GPIO bits. The modulation signal is priduced by a filtered PWM channel from the PI.

4 Raspberry PI Configuration Steps

These are the steps to initially bring up the Raspberry-PI ZERO-W.

This is needed for the first transmitter. Subsequent transmitters can simply be cloned from the firwst working unit.

4.1 Initial Software Load & Configuration

These steps seem to demand that we be locally connected to the Raspberry PI. A keyboard, mouse and monitor seem to be required. Note that the Raspberry-PI uses a micro-HDMI connector so an appropriate cable or and adaptor is required for this setp.

4.1.1 Download Image

Download Raspbian from raspberrypi.org.
Uncompress the image.
Copy to micro-SD card (smallest comfortable size is 16GB)

sudo dd bs=4M if=2019-09-26-raspbian-buster-full.img of=/dev/sdg conv=fsync status=progress

4.1.2 First Run of PI

Connect Monitor and keyboard. Mini-HDMI adapter may be required.

Micro-USB to USB-A for keyboard/mouse.

Switch on Power.

Expand File System.
The image we copied onto the SD card is a minimum size system. Typically for a 4GB or 8GB card. Use the Raspberry PI utility to expand the file system to fill the entire SD card.

Run *fdisk* and reduce the partition size by about 50MB to 100MB (also see section 4.1.13). This reduction allows the SD card to be imaged after the system has been fully configured and restored to another card without having to deal with a slightly smaller card. Reduce the partition size now so the alteration to the partition table can not affect files (all the free space is at the *far* end of the SD card right now).

Enable Wi-Fi.
For fixed IP address, add the new Raspberry-PI ZERO MAC address to the router static assignment table.

Raspberry Configuration: System

Setting	**Setting**	**Description**
Password		Change it!
Hostname	FOX10	Change to something meaningful
Boot	Desktop	(Change to CLI later)
Auto login	pi	Come up 'running'
Network at Boot	uncheck	NO need to wait
Splash Screen	Enable	(Change to DISABLE later)
Resolution		leave for now
Overscan	Enable	
Pixel Doubling	Disable	

Raspberry Configuration: Interfaces

Service	**Enabled?**	**Description**
Camera	Disable	Not used
SSH	Enable	Remote Management
VNC	Enable	Remote Management
SPI	Enable	Frequency Synthesizer
I2C	Enable	Time Chip
Serial Port	Enable	CI-V Network
Serial Console	Disable	CI-V Network
1-Wire	Disable	No 1-Wire Peripherals
Remote GPIO	Disable	Stand-alone Operation

Raspberry Configuration: Performance

No changes

Raspberry Configuration: Localization

Router information here:
/etc/wpa_supplicant/wpa_supplicant.conf

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=US
network={
ssid="<ssid>"
scan_ssid=1
psk="<passphrase>"
key_mgmt=WPA-PSK
}
```

Change *ssid* and *passphrase* as needed.

Raspberry Configuration: Update O/S.

As the PI-ZERO is a single core processor, this step will take some time.

4.1.3 VNC

Access the Raspberry-PI using a VNC client.

Continue with configuration and application software load.

4.1.4 Hardware Clock

The following description assumes the battery is populated on the circuit board and that the DS1672 is operating normally. If this is not the case, it probably won't work as described.

The DS1672 has a clock driver as part of the Raspberryu-PI raspbian distribution.
It is also supported by "i2c-tools" which should be present as part of the raspbian distribution.

One of the following can be used to start the DS1672 and configure the trickle charge function **before** the rtc-ds1672 module is loaded.

```
sudo i2cset -y 1 0x68 0x00 0 0 0 0 0x08 0xA9 i \# 250 ohm resistor trickle charger
sudo i2cset -y 1 0x68 0x00 0 0 0 0 0x08 0xAA i \# 2K ohm resistor trickle charger
sudo i2cset -y 1 0x68 0x00 0 0 0 0 0x08 0xAB i \# 4K ohm resistor trickle charger
```

Dump the DS1672 to confirm that bytes 4 and 5 were written. Bytes 0..3 are the clock registers and will increment every second.
Note that bit 7 of register 4]textbfmust be cleared for the DS1672 oscillator to run.

```
sudo i2cdump -r 0x00-0x05 -y 1 0x68 c
```

Now that the DS1672 oscillator id running and the trickle charger is enabled, we can load the driver and carry on with normal operation.

```
modinfo rtc-ds1672
sudo modprobe rtc-ds1672
```

This is required to cause */dev/rtc* and */dev/rtc0* to appear in the */dev* directory.

```
sudo echo ds1672 0x68 > /sys/class/i2c-adapter/i2c-1/new_device
```

Now we can load the DS1672 with the system time. This assumes, of course, that we have managed to get the time correctly set.

```
sudo hwclock -w
```

We should see it echo back what we write to it.

```
sudo hwclock -r
```

Remove fake hardware clock

```
sudo systemctl disable fake-hwclock
```

If that all seems to work, then we can patch it into the init scripts to get time set from DS1672 during boot.
The local init script */etc/rc.local* is convenient. We need not get too facny about where we add this as it will run late in the boot process.

```
#!/bin/sh -e
#
# rc.local
#   logs to /var/log/messages
#
# 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.
#
#  Enable the DS1672 and read it
#
logger "DS1672  0 ************************************************************"
logger "DS1672  1 /etc/rc.local"
logger "DS1672  2      i2c modules: `grep i2c /proc/modules`"
logger "DS1672  3      rtc modules: `grep rtc /proc/modules`"
#
logger "DS1672 10 cmd: modprobe i2c-dev          ; load device driver module"
modprobe i2c-dev
#
#  Battery will keep this set ?
#
logger "DS1672 20 cmd: i2cset -y 1 0x68 4 0x00 0xAA i ; enable oscillator"
i2cset -y 1 0x68 4 0x00 0xAA i
#
logger "DS1672 21 cmd: modprobe rtc-ds1672       ; replace the clock module"
modprobe rtc-ds1672
#
logger "DS1672 20      i2c modules: `grep i2c /proc/modules`"
logger "DS1672 21      rtc modules: `grep rtc /proc/modules`"
logger "DS1672 22      dev:         `ls -l /dev/rtc*`"
logger "DS1672 23 cmd: echo ds1672 0x68 > /sys/bus/i2c/devices/i2c-1/new_device"
echo ds1672 0x68 > /sys/bus/i2c/devices/i2c-1/new_device
#
logger "DS1672 30 cmd: hwclock -s (Set  the System Clock from the Hardware Clock.)"
hwclock -s
#
# Print the IP address
#
logger "DS1672 50 My host name is  `hostname`"
logger "DS1672 51 My IP address is `hostname -I`"
#
exit 0
```

4.1.5 Audio and Sound Player

Simple Audio Player

```
sudo apt-get install raspi-gpio
gpio -v
gpio readall
sudo apt-get install sox
sudo raspi-config
        7. Advanced Options\\
        A4. Audio//
        1. Force 3.5mm\\
amixer cset numid=1 100%
alsamixer
```

Change /boot/config.txt looking for the area like below and add the gpio and dtoverlay lines.

```
# Additional overlays and parameters are documented /boot/overlays/README
# Enable audio (loads snd_bcm2835)
gpio=12,13=op,a0
dtoverlay=pwm-2chan,pin=12,func=4,pin2=13,func2=4
dtparam=audio=on
```

4.1.6 cwwav

Morse Code Generator
Runs on the host system as it requires severeal support libraries.

This routine generates morse code from a text file. The output is a .wav file that is played on the Raspberry-PI ZERO.

4.1.7 GPIO from Wiring PI

Wiring PI provides a command line utility to manipulate the GPIO pins. Although it is mentioned here, it is not really necessary.

Useful for low level hardware testing.

`http://wiringpi.com/the-gpio-utility/`

4.1.8 PI GPIO library

This library provides an interface that can be accessed from C. The utility routinesd for the fox transmitter use this library.

```
http://abyz.me.uk/rpi/pigpio/
```

download & install

```
rm pigpio.tar
sudo rm -rf PIGPIO
wget abyz.me.uk/rpi/pigpio/pigpio.tar
tar xf pigpio.tar
cd PIGPIO
make
sudo make install
```

4.1.9 daemontools

Tools to copy SD card images. See section 4.1.13.

```
sudo apt-get install daemontools daemontools-run
```

4.1.10 Apache 2

This library provides a web browser interface to monitor the Fox Transmitter.
download & install

```
$ sudo apt install apache2 -y
```

4.1.11 SAMBA

Install SAMBA to make access to remote file systems a bit more convenient.

```
sudo apt-get install  samba-common smbclient samba-common-bin smbclient  cifs-utils
```

4.1.12 Power Minimization

Change the wat the LED is handled.

```
echo gpio | sudo tee /sys/class/leds/led0/trigger
```

Turn the LED off (yes it is backwards).

```
echo 1    | sudo tee /sys/class/leds/led0/brightness
```

Turn the LED on.

```
echo 0    | sudo tee /sys/class/leds/led0/brightness
```

Turn off HDMI section.

```
sudo tvservice -o
```

4.1.13 SD Card Cloning

Notes for cloning an SD card

use dd to copy use parted to resize to slightly smaller
edit /etc/hosts ¡- change local host line and update FOX Tx list
edit /etc/hostname ¡- change fox name
update router

5 FOX Transmitter Utilities

Utility routines that allow implementing the fox function on the Raspberry-PI

These provide convenient access to the hardware and some timing support functions.

5.1 AD7991

Depends on PIGPIO or PIGPIOD.

This is the routine that deals with the AD7991 A/D subsystem.

5.1.1 -h

Help Text

5.1.2 -s

Output stream for *vocalizer*

5.1.3 -b

Battery, Volts and counts 1 fractional digit

5.1.4 -B

Battery, Voltage only

5.1.5 -i

Battery, Current milliAmps and counts 1 fractional digit

5.1.6 -I

Battery, Current only

5.1.7 -r

Voltage & Current with timestamp to FOX_Status_File

5.1.8 -R

Voltage & Current to AD_xMON_RF in FOX_Status_File

5.1.9 -c

Photocell, Volts and counts 1 fractional digit

5.1.10 -C

Photocell, Volts only

5.1.11 -d *n*

Wait for Photocell to drop below nV

5.1.12 -D*n*

Wait for Photocell to rise above nV

5.1.13 -m *n*

Wait for Photocell to change more than nV

5.1.14 -N *name*

name to FOX_Status_File

5.1.15 -S *n*

Sleep n seconds.
Keep A/D data in FOX_Status_File up to date

5.2 cgi-bin

Depends on n/a.

This provides the code that is executed to render the fox status web page.

This also provides a means of setting fields on the web page from worker programs.

5.2.1 Interactive

Three arguments.

argument 1 is always the keyword **SET**. An incorrect keyword here causes the entire command to be ignored.
argument 2 is the key name in the FOX_Status_File
argument 3 is the key value in the FOX_Status_File

Using this routine allows setting fields from a shell script in the FOX_Status_File that are then displayed on the web page.

5.2.2 cgi script(web)

NO arguments.
Provides web formatted *html* page.

5.3 ICS307

Depends on PIGPIO or PIGPIOD.

This routine provides control of the ICS307 clock synthesizer and a few discrete RF section control bits.

Calculating the divisors takes quite a bit of time. To speed things up the ICS307 utility is run once at ther start to calculate the divisors and then these divisors are used thereafter to set the ICS307:

F144250K=‘ICS307 -F144.250 -c 15 -f16K‘

The $F144250K variable now has the divisors that are subsequently used to configure the ICS307.

5.3.1 -q

Quiet.
Suppress logfile output.

5.3.2 -D

INcrease debuf level.

5.3.3 -b *n*

Beep! *n* milliseonds.

5.3.4 -F

Desired Frequency(MHz).

5.3.5 -X

Crystal Frequency(MHz).

5.3.6 -P

Power Ebable.
A Audio subsection **R** RF subsection **P** Assert(ground) PTT pin **D** Random assert DIAG_BUZZ

5.3.7 -l *filename*

List register patterns to *filename*.

5.3.8 -L *filename*

Limited register patterns to *filename*.

5.3.9 -L *n*

n.

5.3.10 -r *n*

Set Reference Divivisor *n*.

5.3.11 -v *n*

Set VCO Divisor *n*.

5.3.12 -c *n*

Set Charge Pump to *n*.

5.3.13 -f *n*

Set Loop Filter Resistor to *n*.

5.3.14 -d *n*

Set Output 1 and 2 Divisors to *n*.

5.3.15 -1

Enable clock 1.

5.3.16 -2

Enable clock 2.

5.3.17 -3

Enable clock 3.

5.3.18 -p

Power up ICS307 (GPIO 17, pin 15).

5.3.19 -a

Power up RF Amplifier (GPIO 16).

5.3.20 -x

Power up Oscillator (pin 16).

5.3.21 -C *callsign*

Set callsign *callsign* in FOX_Status_File.

5.3.22 -N *name*

Set unit *name* in FOX_Status_File.

5.3.23 -N *title*

Set unit *title* in FOX_Status_File.

5.4 Radio

Depends on n/a.

Standalone utility routines.

5.4.1 halo_term

This is a basic terminal emulator for accessing the zNEO based fox transmitters. It uses an alias table to find the USB device and set the communications parameters correctly.

5.4.2 fox_simple

This is a routine that is used to load the FRAM in the zNEO fox transmitter through the USB port.

5.4.3 schedule

This is a routine that is called from a script to control scheduling.

This may ne called to perform MOD scheduling.

This may ne called to perform timed scheduling.

5.4.4 scsim_align

This is an implementation of the MOD function that is used to effect scheduling.

5.4.5 net_time

Depends on CI-V network connection.

This is the time utility for the **Fox Transmitter network**.

In master-mode, the utility sends a single time message out with each invocation. This message occurs 100mS before the seconds field in the system time changes. The message content is for the next second. See section 5.4.7.

The receiving station decodes the message and updates its system time which aligns the clocks to within a few tens of milliseconds. This is **not** an attempt to achieve the precision of **NTP**.

The format of the time message is the same for all Fox Transmitters. The time in the message is truncated to cover a 10-day span. Received time message traffic is further truncated to a single day and updates the local time without altering the date.

ZULU Message Format
ZULUSP **1234**CR

The 4 character key string **ZULU** identifies this as a GMT/UT truncated time message. The **1234** is the current time expressed in seconds and truncated to a small number of days.

In master-mode a single message is sent to allow the controlling script to determine the message cadence. The *-M sec* flag set master-mode and establishes the scheduling synchronization point for the outgoing message. Nominally the *sec* field would be set to 1 or 2.

In slave-mode, the utility continuously listens for time traffic on the fox time network. When a *ZULU message* is found, is is used to update the local system clock.
As mention earlier, the update changes only the second-of-day, not the year, month, or day.

5.4.6 -S *device*

Network Time *device*.
S0 selects /dev/ttyS0
AMA0 selects /dev/ttyAMA0

5.4.7 -M *seconds*

Time broadcast synchronization period in *seconds*
This flag sets the MASTER mode in which time is sent over the network.
If this flag is **mot** set, the routine operates in slave mode.

5.4.8 -D

Network time message is sent to stdout (diagnostic).

5.4.9 -d

Increase debug level

5.5 Readsw

Depends on PIGPIO or PIGPIOD.

This utility routine reads the switch bits.

5.5.1 -m

Master Jumper installed 1/0.

5.5.2 -M

returns MASTER/SLAVE.

5.5.3 -t

Test Jumper installed 1/0.

5.5.4 -T

returns TEST/NORMAL.

5.5.5 -s

Talkie Toaster Switch.

5.5.6 -b

Switch (synonym) closed 1/0.

5.5.7 -S

returns ACTIVE/IDLE.

5.5.8 -B

(synonym for -S).

5.5.9 -a

Diagnostic Switch: All Jumpers

5.5.10 -r

Diagnostic Switch: RAW pin status

5.5.11 -N *name*

name to FOX_Status_File

5.6 Servo

Depends on PIGPIO or PIGPIOD.

Utility routine to control the R/C servo channel.
Talkie Toaster support routine.

5.6.1 -q

Quiet!
Stop servo pulses.

5.6.2 -v

Verbose, talk about what you're doing

5.6.3 -p *percent*

Pulse width *percent*.

5.6.4 -w *usec*

Pulse width *usec*. 1000..3000.

5.6.5 -a *degrees*

Angle in *degrees*. 0..180.

5.6.6 -s *n*

STALL for *n* seconds.

5.6.7 -r *seconds*

random movement, duration in *seconds*.

5.6.8 -t *sec*

Test pin function (square wave for *sec* seconds).

5.6.9 -N *name*

name to FOX_Status_File

5.7 vocalizer

Depends on voice files in WAV directory.
Depends on /usr/bin/play to send audio files to audio device.

Utility routine to vocalize status messages.

5.7.1 phrase_table_file_name

`/home/pi/vocalizer/phrase_table.csv`

Translation table for text fragment to sound file.
Files are located in the WAV directory.

When the characterr group in the first colum in the phrase table file is encountered, the .wav file in the second column is played.

Order of the file is important, once a character group is matched it is played and the decoder moves on to the next character. Larger groups must appear first in the file.

If matching fails here, the vocalization continues with the internal tables listed below.

5.7.2 base vocalizations

These vocalizations are located in the WAV directory.

When the character in the first colum is encountered, the .wav file in the second column is played.

```
  0    V_N0.wav
  1    V_N1.wav
  2    V_N2.wav
  3    V_N3.wav
  4    V_N4.wav
  5    V_N5.wav
  6    V_N6.wav
  7    V_N7.wav
  8    V_N8.wav
  9    V_N9.wav
```

5.7.3 base vocalizations

These vocalizations are located in the vocalizer directory.

When the characters in the first two colums are encountered, the .wav file in the last column is played.

```
P p    photo.wav
C c    cell.wav
B b    battv.wav
I i    batti.wav
  m    milli.wav
A a    amp.wav
V v    volt.wav
  .    point.wav
```

5.8 WAV

Depends on n/a.

Data files for the vocalizer and other sound files used by the fox transmitter.

6 FOX Transmitter Scripts

A set of shell scripts operate the fox transmitter are descriped here.

The fox system is physically controlled by the various utilities in section 5. The order of operation is controlled by these shell scripts.

6.1 FOX.bash

Called by:
System Initialization

Calls on:
Soft link to the main shell script.

Description:
Use a soft link to make changes easier.

6.2 fox_transmitter.bash

Called by:
FOX.bash

Calls on:
fox_transmitter_entry.bash
FOX_ID.bash
FOX_Alias.bash

Description:
Main control loop.

The script call on FOX_ID.bash and FOX_Alias.bash to set all of the variables that will be used by the scripts. The unique identity of the fox transmitter is controlled by the hostname assigned to the unit in the /etc/hosts file.

6.3 fox_transmitter_entry.bash

Called by:
fox_transmitter.bash

Calls on:
FOX_ID.bash
FOX_Alias.bash
Arguments:
$1: ICS307 setup string $2: Loop COunt

Description:
Send Single Message.

The setup for the ICS307 include the divisors and the VCO control bits.

The loop count is a simple count, used for debugging.

6.4 FOX_ID.bash

Called by:

Calls on:

Description:
Set the CALLSIGN and UNIT NAME.
UNIT NAME comes directly from the hostname.
CALLSIGN must be edited.

The operating schedule is set in this script. It is sensitive to the $UNIT variable to select unique offsets into the scheduling period for each unit.

6.5 FOX_Alias.bash

Called by:

Calls on:

Description:
Set variable names for executables and audio files.

Full path to the utility routines are defined here so other shell scripts have easy access. Changes to the locations of the utility files is handled here.

Some of the audio files are unit specific and are tailored here.

6.6 cfg_transmitter.sh

Called by:

Calls on:

Description:
Sets up the FOX.bash symlink so it points to fox_transmitter.bash

6.7 fox_time_master.sh

Called by:
fox_transmitter.bash

Calls on:

Description:
Sends *ZULU message* out on the time network (CI-V)

6.8 fox_time_slave.sh

Called by:
fox_transmitter.bash

Calls on:

Description:
Waits for incoming *ZULU message* traffic from the time network (CI-V)